TAMPERE UNIVERSITY OF TECHNOLOGY

EMMA RINNEVAARA

ALTERNATIVE BUSINESS MODELS FOR INDUSTRIAL SOFTWARE SERVICE SOLUTIONS

Master of Science Thesis

Prof Miia Martinsuo has been appointed as the examiner at the Council Meeting of the Faculty of Business and Built Environment on October 8th, 2014.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Industrial Engineering and Management

RINNEVAARA, EMMA: Alternative Business Models for Industrial Software Service Solutions
Master of Science Thesis, 94 pages, 5 appendices (6 pages)
November 2014
Major: Industrial management
Examiner(s): Professor Miia Martinsuo
Keywords: Business models, industrial software, on-premises software, software-based services, Software as a Service

Industrial firms need software solutions to coordinate and control their systems. This thesis focuses on comparing three alternative business models for industrial software service solutions. The three software business models are the on-premises software, software-based services and Software as a Service. On-premises software is run on the customer's servers and it has with three alternative revenue models: the perpetual license, maintenance agreement and the subscription model. The software-based services are services created from the outputs of software. Software as a Service is run the vendor's servers and accessed over the internet, the business model has three main revenue models: subscription, pay-per-use and freemium model.

On-premises software is an old concept but for instance the software-based services is a model completely missing from literature. Also the industrial point of view is absent from the Software as a Service literature. The thesis begins with a literature review but to discover the industrial perspective on the topic, five focus group discussions, with software and industry participants, were conducted. Based on these discussions a framework was constructed. The framework was then refined by conducting four unstructured interviews with key experts and finally the framework was validated by a weak market test in two workshops.

The framework, constructed as the result of the thesis, consists of two decision trees to discover the best fitting delivery model and revenue model based on set specifications of a software product and the targeted customer in an industrial setting. The software vendor in this thesis is presumed to be an engineering company. The delivery model presents the obstacles for using an external server to run the software, the opportunities that the external server presents, the restrictions that may influence the decision between the customer's and vendor's servers and the final three delivery models. The revenue model decision tree presents a path where the two starting points are the on-premises software and the delivery model for Software as a Service. The framework does not present the most profitable and cost effective business models but the best fitting, based on distinguishing requirements and opportunities of each software business models.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tuotantotalouden koulutusohjelma

RINNEVAARA, EMMA: Teollisten palveluohjemistojen vaihtoehtoiset liiketoimintamallit
Diplomityö, 94 sivua, 5 liitettä (6 sivua)
Marraskuu 2014
Pääaine: Teollisuustalous
Tarkastaja: Professori Miia Martinsuo
Avainsanat: Liiketoimintamallit, teolliset ohjelmistot, perinteiset ohjelmistot, ohjelmistopohjaiset palvelut, ohjelmistot palveluna

Teolliset yritykset tarvitsevat ohjelmistoja prosessiensa kontrollointiin ja koordinointiin. Tämä diplomityö vertailee kolmea vaihtoehtoista liiketoimintamalli teollisille palveluohjelmistoille. Käsiteltävät liiketoimintamallit ovat perinteiset ohjelmistot, ohjelmistopohjaiset palvelut sekä ohjelmistot palveluna. Perinteiset ohjelmistot toimivat asiakkaan palvelimilla, ja niiden mahdollisia ansaintalogiikoita ovat kertaluontoinen lisenssi, ylläpitosopimus sekä tilausmalli. Ohjelmistopohjaiset palvelut ovat palveluita, jotka muodostetaan ohjelmistojen tuotosten perusteella. Ohjelmistot palveluina toimivat toimittajan palvelimilla ja asiakas käyttää niitä netin välityksellä. Tähän liittyvät ansaintalogiikat ovat tilaus-, maksu käytön mukaan - sekä freemium-malli.

Perinteisiä ohjelmistoja on tutkittu laajasti, mutta ohjelmistopohjaisia palveluita ei löydy käsitteenä kirjallisuudesta ollenkaan. Ohjelmistot palveluina -liiketoimintamallin osalta kirjallisuudesta sen sijaan puuttuu teollisuusohjelmistojen näkökulma. Työ alkaa kirjallisuuskatsauksella, mutta teollisuuden näkökulman saamiseksi toteutettiin viisi fokusryhmäkeskustelua. Keskustelujen perusteella luotiin viitekehys, jota jalostettiin neljän strukturoimattoman asiantuntijahaastattelun avulla. Tämän jälkeen suoritettiin heikko markkinatesti kahdessa työpajassa viitekehyksen validoimiseksi.

Tuloksena muodostunut viitekehys koostuu kahdesta päätöspuusta, jotka määrittävät sopivimmat toimitusmallit sekä ansaintalogiikat ohjelmiston ja kohdeasiakkaan ominaisuuksien perusteella teollisessa ympäristössä. Toimintamallipäätöspuu koostuu neljästä alaryhmästä, jotka määrittävät esteet, mahdollisuudet sekä rajoitukset, jotka liittyvät ohjelmiston pyörittämiseen toimittajan palvelimilla ja neljäntenä ryhmänä ovat toimitusmallit. Ansaintalogiikkapäätöspuu määrittää sopivimman ansaintalogiikan perinteisten ohjelmistojen sekä ohjelmistot palveluina toimintamalleihin. Tulos ei kerro kannattavinta tai kustannustehokkainta ratkaisua, mutta se tuo esiin tarkasteltavien liiketoimintamallien edellytykset ja mahdollisuudet.

PREFACE

This thesis has been conducted on behalf of ABB Group Service in Zürich Switzerland. During my time working with the Group Service team I have learned a lot. In this team I have been privileged to work with wonderful and unbelievably skilled and smart people. I owe all of you a big thank you for the support and help you have given me. You have made this an unforgettable experience. I am sure to miss every single one of you and I hope to see all of you again someday.

I have to extend a special thank you to Christopher Ganz who acted as my supervisor for the thesis and proved to be a wonderful help to get me started with the study, to Jari Kaija for giving me this opportunity, for Zied Ouertani for helping and advising me when I felt like I had hit the wall with my thesis. I would also like to thank Professor Miia Martinsuo for helping me find and keep to the right track with my thesis.

I also owe a big thank you to my boyfriend Niklas who stood by me despite the distance and endured with my stress when I needed to let it all out. I would also like to thank my parents for believing in me and especially my mom for trying her best to carry my stress and worries for me through this experience, just as any experience in my life. Thank you to my sister Anna who came to brighten up my stay in Zürich. Thank you to Claudia without whom my time in Zürich, writing this thesis, would not have been the same. Thank you for the great talks and adventures that have given me energy and wonderful memories.

This thesis concludes an important time in my life. I would like to thank all my friends for making the years in university the best years of my life so far. I hope there are many more amazing times we can experience together. Also thank you to everyone who has supported me throughout all my school years. I owe this moment to all of you.

Zürich, 27.11.2014

Emma Rinnevaara

TABLE OF CONTENTS

1. INTRODUCTION

New emerging business models in the software industry are appearing in the market, an example of which is the concept of Software as a Service. These emerging trends are of interest not only in the consumer markets but also in the industrial world. ABB is one of those companies wanting to look into the new business models in the industrial setting. In order to get a broader view in to the new trends, they should be looked at in comparison to the already existing and more well-known business models.

1.1. Research background

Industrial software on its own seems to be a term without a clear definition in the literature. Due to this there are also no specifications to the industrial software business models. However, from literature and the industry three software business models interesting to this thesis can be identified. The three software business models considered are the on-premises software, the software-based services and Software as a Service.

On-premises software is the most traditional software offered in the market up to date. In this kind of software usually a permanent license is bought and the software is installed on the computers of the customer. (Xin & Levina 2008; Waters 2005.) As the software is located on the customer's servers, also the maintenance of the software is regarded as the responsibility of the customer (Xin & Levina 2008). Most of the software systems mentioned in the case studies about industrial software are on-premises software, even though this is not specifically defined in the studies.

Software-based services is a concept that cannot be found in literature as is. It has a lot of aspects defined for software-enabled services by Black (2008) and various articles on e-services. These studies however do not explain the concept of software-based services fully. The basis of the software-based services is that the software that is run on the vendor's servers and the outputs of this software are used to create a software offering that is then delivered to the customer. The customer may or may not have an interface to this software, but the most important factor is that what the customer pays for is not the outputs of the software itself but rather the services derived from the outputs.

Software as a Service is seen as the new delivery method for software (Waters 2005). What is relatively new in this is the idea that the software is run on the servers of the vendor (Choudhary 2007; Ojala 2013; Waters 2005). While this has been an emerging

subject of study in the recent years the focus of the research has mainly been in the consumer software or enterprise software. In the industrial setting the traits of Software as a Service are looked into very scarcely, if at all.

1.2. Research questions and objectives

The objective of this thesis is to view the different industrial software business models in comparison to each other. The different business models under examination are the traditional on-premises software, software-based services and Software as a Service. The main research question to be answered in this thesis can be stated as follows:

> *What aspects differentiate the alternative business models for industrial software service solutions?*

In other words the main objective of the thesis is to figure out what are the things that distinguish the different business models from each other. The main research question breaks down to the following three questions:

> *What is the nature of each business model: on-premises software, delivering software-based services and Software as a Service?*
>
> *What are the things that an engineering firm needs to consider when choosing one of these business models?*
>
> *What opportunities do the business models bring forth to an engineering firm?*

In order to find the differentiating traits the three business models need to be understood. The things that need to be considered when choosing a business model is one way to find differentiating traits but in addition the different opportunities of each business model are also qualities that differentiate them from each other.

In the academic sense the thesis aims to contribute to the literature on the business models for industrial software and especially the aspects that make these business models different from each other. What are the opportunities and benefits in them and when is it beneficial to use each. In regards to ABB the objective of the study is to shed light on the alternative business related to industrial software and its service solutions. Especially interesting is to find out if through Software as a Service new opportunities can be reached, that cannot be seized through the traditional software business models, as well as see which business models should be looked into further.

1.3. Research context

This thesis is conducted on behalf of ABB Group Service in Zürich, Switzerland. Service and software will play a big role in the new ABB 2020 strategy. Software as well as service are both based on providing knowledge and knowhow to the customer. Software as a Service is an intriguing linkage between the two and therefore falls within the scope of ABB Group Service.

This thesis is carried out in regards to a new strategy project concerning Software as a Service and other service related software options as possible new business ventures. The aim of the larger project is to look into possible business models in the respective scope as a means to broaden the ABB service offering and eventually support growth.

The function of Group Service in ABB is to serve with a strategic purpose for the service business in ABB globally. As the thesis is conducted on behalf of ABB Group Service and with ABB being a multi-industry organization operating in power and automation the scope of this study needs to be kept on a somewhat general business model level, without focusing on any industry or product in particular.

Due to the research being conducted on behalf of a multi-industry engineering company and especially since the thesis is conducted in the headquarters of the company, aiming for the scope of the entire firm, some delimitations come to play. The thesis will not go into detail on the actual software products as they may vary across the industry and business unit vastly. This way a less granular approach on the business models can be reached within the scope of ABB. The setting also focuses the study to concentrate on an engineering firm acting as the vendor rather than a software company.

Conducting the thesis for Group Service also outlines the scope to software service solutions. While the distinction to regular software solutions is not big, the service aspect does seem to bring a more data driven approach to the business models. The service aspect is reached by focusing the empirical study on the service and software business of ABB.

1.4. Research methods

After looking into literature on the subject the industry point of view was gathered through focus group discussions. These focus groups were comprised of ABB employees from various areas and stages in the organization. All in all the participants comprised quite a good representation of the entire organization, from participants from various business units to members from the corporate research teams and global strategy team. All the participants had some insight into the software industry and software business models from the operations or on a theoretical level. The aim of the focus

group discussions was to brainstorm about the different business models and the things that would set them apart from each other.

From the results of these interviews, an initial framework was comprised. This framework was utilized in the refining supplementary interviews. These were unstructured interviews with a select group of experts from the organization. The participants had better knowledge of the different business models and therefore more insight on the matter. During these interviews the initial framework was examined step by step initiating feedback and questions from the interviewees. Thought encouraging questions were asked on their opinions on different sections of the model. Based on this interview round, the framework was refined by introducing some changes to it.

The framework was then validated through a weak market test. The weak market test was conducted by organizing two final workshops where the final version of the framework was presented. The first workshop consisted of the team members from Group Service and the second of almost all the participants from the previous focus groups and interviews. The frame work was open for discussion and critique during this session.

1.5. Structure of the thesis

The thesis is divided in to six chapters. The first chapter entails the literature review looking in to business models, industrial software, on-premises software, software-based services and Software as a Service. The aim of the literature review is to get a good idea on the setting of the thesis as well as understanding the central concepts. The literature review also functions as the theoretical foundation for the empirical part of the study.

After the literature review the research method is introduced. This part explains the methodologies used to build the empirical part of the study. The chapter presents the research strategy, the methods for data collection and the ways that the data was then analyzed.

The fourth chapter presents the results from the empirical data collection and analysis. In this thesis the framework is introduced in two parts through the delivery model and the revenue model parts of the framework. The incremental changes made to the framework are seen after this in the subchapter titled “Refining of the framework in the supplementary interview rounds”. The final subchapter introduces the results from the validation process and summarizes the final framework.

The fifth chapter is the discussion chapter. This chapter analyzes the results of the study. The chapter looks into the opportunities and challenges of the framework and also proposes the next steps for the company.

The sixth chapter is naturally the conclusion of the thesis. The conclusion chapter examines the academic contributions and the managerial implications of the study. It also examines how the objectives of the thesis were met and discusses the limitations of the research. Lastly, in the sixth chapter the opportunities for future research are introduced.

2. LITERATURE REVIEW

The literature review sets the basis for the thesis. It introduces the setting in which the research is done and examines the different aspects related to the thesis from the literature point of view. First the concept of business model is defined and the relevant components are introduced. After this, in the second subchapter, the term industrial software is outlined and therefore the basis of the scope of the thesis is set. In the third subchapter the on-premises software is looked at from the point of view of the delivery model and the associated revenue models. The software-based services is defined in the fourth subchapter following the examination of the delivery and revenue models of Software as a Service in the fifth subchapter. The three previously mentioned subchapters will act as the main foundation for the empirical study later. Finally, the literature review is summarized and the concepts looked at together as a whole in the sixth subchapter.

2.1. Business model

In literature there seems to be no generally recognized definition for the term business model, it is often used interchangeably with strategy, business concept, revenue model or economic model (Morris et al. 2005). Teece (2010) summarizes the business model to be the way how to create value to the customer, how to deliver it to the customer and how to make profit out of it. Chesbrough & Rosenbloom (2002) also insinuate the business model having a connection between technical potential and the realization of economic value.

A lot of articles describe business models in varying ways. Often the business model has been divided into components that bring together the different aspects that business models should consider. Shafer et al. (2005) constructed their model by comparing previous research and found 42 different components. From these components four categories were named: strategic choices, creating value, capturing value, and the value network. (Shafer et al. 2005.) Morris et al. (2005) discovered three levels of business models the first being the basic components. The six presented basic components were phrased as six key questions to characterize the business model. (Morris et al. 2005.) Chesbrough & Rosenbloom (2002) also list six different functionalities for business models. Tsvetkova & Gustafsson (2012) have created a four component model based on previous literature that summarizes the main components of the business model and the component’s relations to each other. The Morris et al. (2005) and Chesbrough &

Rosenbloom (2002) components can be interpreted to fit under the four components introduced by Tsvetkova & Gustafsson (2012).

The first component in the Tsvetkova & Gustafsson (2012) model is value proposition. This means the benefit that the customer receives through the product or service that is offered and the benefit is based on the perception of value by the customer. (Tsvetkova & Gustafsson 2012.) Casadesus-Masanell & Ricart (2010) describe the business model as the logic behind the firm's operations to create value to its stakeholders. This is also backed by Chesbrough & Rosenbloom (2002) as they list one of the functions of the business model to be the articulation of the value proposition and specify this value as the value that is perceived by the customer. Value creation is also one of the components in the model formed by Shafer et al. (2005). They perceive value creation to include the core competencies, capabilities and positional advantages that ensure the additional value to the customer (Shafer et al. 2005). Also Morris et al. (2005) state the question of how the firm creates value as a key question in the basic components of business models.

Tsvetkova & Gustafsson (2012) define one of the business model components to be the capabilities, they describe the capabilities as the factors that define how the value is delivered to the customer. The components of value creation in the studies of Morris et al. (2005) and Shafer et al. (2005) link the value proposition and the capabilities needed to deliver this value proposition together. Morris et al. (2005) also note that defining the internal source of advantage should be considered as one of the key questions when characterizing a business model. Tsvetkova & Gustafsson (2012) point out that capabilities can be described through the network of suppliers and partnerships. This aspect is backed by quite a few other studies. Amit & Zott (2001) state that the business model should define how advantage is gained over the competitors. Morris et al. (2005) also refer to this when they describe one of the questions behind business models should be how the company is positioned in the market place. Also Chesbrough & Rosenbloom (2002) describe the structure of the value chain as well as the position of the focal firm within a value network as important aspects in the business model. Shafer et al. (2005) in fact account the value network to be one of the main components of the business model.

Through the capabilities the value can be delivered to the customer, but as the value is defined through the customer perception, the customer itself is a key element in the business model (Tsvetkova & Gustafsson 2012). Also the relationship with the customer and the information about the customer links the value chain aspect of the capabilities to the customer component (Shafer et al. 2005). Morris et al. (2005) define the question for who to create value to as the first key questions of the business model. This includes the nature and scope of the market as well as the customers' characteristics. The identification of the customer segment is a key function of a

business model (Chesbrough & Rosenbloom 2002). The customer types and their requirements when it comes to interaction, organizational configurations and their dispersion geographically can make a difference in the business model (Morris et al. 2005).

The revenue model shows how the money flow forms from the capabilities, through the value proposition to the customer (Tsvetkova & Gustafsson 2012). The revenue model is stated in almost every study about the compiling of a business model. According to Sainio & Marjakoski (2009) the revenue model describes how revenue is collected from customers or partners. Teece (2010) states that the business model holds assumptions about costs and revenues. Morris et al. (2005) also define revenue sources, cost structures, pricing methodologies, margins, and expected volumes in relation to the important question of how the company will make money, that should be answered by the business model. Shafer et al. (2005) describe the way for the firm to make money as value creation but this can be interpreted to mean the same as the revenue model. Teece (2010) compiles the business model to shape the architecture of the costs, revenues and profits that are related to creating said value and Chesbrough & Rosenbloom (2002) agree that the structure of the cost structure and the profit potential are an essential part of the business model. All in all, the revenue model can be deemed an essential component of the business model stated almost unanimously in the current literature.

Tsvetkova & Gustafsson (2012) note that the four components: value proposition, capabilities, customer and the revenue model are all closely related to each other and can only be separated for analytical purposes for example to distinct how business models differ from one another. One of the combining factors of the four elements could be seen as what Shafer et al. (2005) portray as their first category of business model components, the strategic choices. Meaning choices concerning the competitive advantage of the company such as customers, competitors, revenue models and branding to name a few. (Shafer et al. 2005.) This category includes elements from all of the four business model components and form a strategic bond within them. Also Morris et al. (2005) discuss that while the business model is not the same as a business strategy it does hold a number of strategic elements within. Teece (2010) reminds that the business model should involve assumptions about the customer behavior, likely competitor, revenues as well as the changing nature of customer needs, these could well be seen as strategic elements in the business model. The business model should also be assessed against the changing business environment and ecosystem (Teece 2010). Morris et al. (2005) also consider the time, scope and size ambitions behind the business model, which could be thought of as strategic decisions. Chesbrough & Rosenbloom (2002) describe the formulation for the competitive strategy as one of the functions of the business models. All things considered, the strategy aspect of the business model can be seen as the ways that the four elements are connected.

While in this thesis these components, the value proposition, capabilities, customer and revenue model, are interpreted as being the main components in any kind of business model more specifically a business model related to a certain product or service it must be recognized that most of the literature examine the business model in the scope of an entire company. For example Tsvetkova & Gustafsson (2012) examine the entire industrial ecosystem, Teece (2010) has the scope of an entire enterprise business model and Morris et al. (2005) also consider the business model to be for the entire company. In the case of this thesis, the value proposition of the entire company cannot be described simply but rather the business model for one product is then assumed to be the same as a business model for a single product company in an industrial ecosystem. Some of the literature also looks into the business models of e-business, examples of these are Shafer et al. (2005) and Amit & Zott (2001) while e-business may come closer to the setting of industrial software that is looked at in this thesis it cannot be said to be exactly the same. Chesbrough & Rosenbloom (2002) study also comes close as they examine business models for new technology. This thesis does include some fairly new technology but the scope is not as narrow in the case of this thesis.

Figure 1 represents the four components of the business model adapted from the Tsvetkova & Gustafsson (2012) model. The four components are named value proposition, customer, revenue model and delivery model. The arrows in the figure represent the fact that all the components are mutually related to each other.

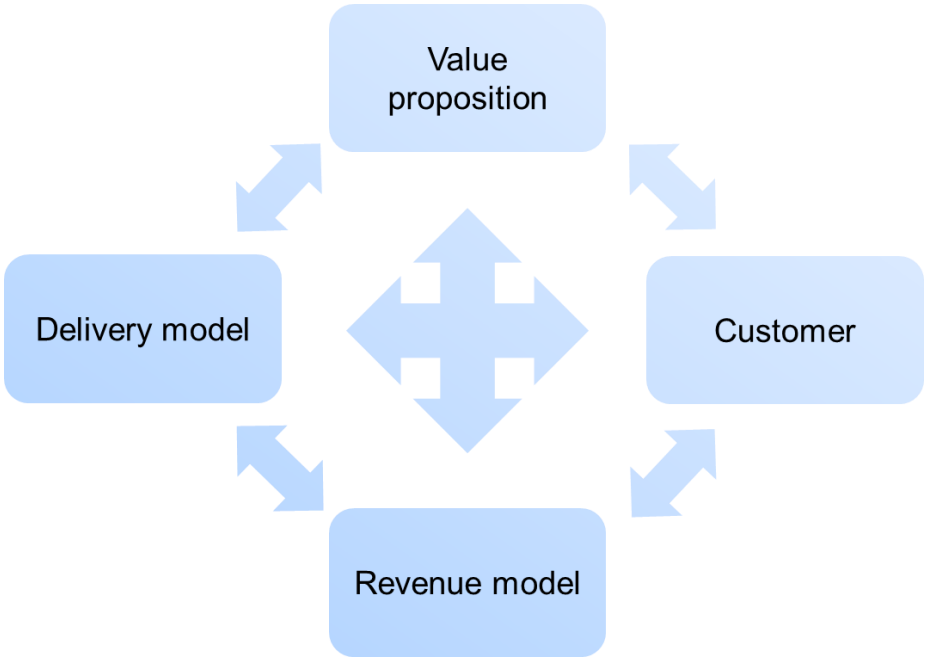


Figure 1. *The business model components (adapted from Tsvetkova & Gustafsson 2012).*

The targeted customer in the Figure 1 is the component that is very specific to the company and the industry where the company operates. The value proposition is very specific to what the company wants to offer the customer. It can then be assumed that when knowing these the best possible way to deliver the value and the best way to generate money in this case can be derived from information from the two previously mentioned components. Also the setting in which the business model is utilized needs to be well defined and this sets limits to the other two components.

The capabilities are depicted in the Figure 1 as the delivery model, because Tsvetkova & Gustafsson (2012) the capabilities represent how the value is delivered to the customer. The setting of industrial software sets limits to what the delivery of the value can be as well as how the money, meaning what the revenue model can be. The fitting delivery model in this setting can be derived from information about the value proposition and the targeted customer and the environment in which the customer operates. In general, it can be assumed that in the industrial software business models the value proposition and target customer in regards to the software can be the same but the business models then differentiate mainly from the delivery and revenue model perspectives.

With the previously mentioned assumption in play in the next chapter will define the setting of this thesis. After this the different delivery models and revenue models within will be examined in the chapters following.

2.2. Industrial software

In literature there seems to be no common definition for industrial software. While the term is widely used in literature no explanations are offered to specify the limits of industrial software. It can however be deduced that industrial software covers embedded software, systems software and application software used in industry, utilities as well as infrastructure and transportation.

Embedded software, systems software and applications software also lack distinct definition. Nonetheless, some defining aspects can be found within literature of different aspects of industrial software. For example, van der Linden et al. (2009) define embedded software systems as systems where the software is not the main value bringing component in the customer's point of view.

Also systems software can be embedded. Jaffe et al. (1991) give process control systems as an example of systems software used in the industry. They state that control systems control large physical systems and are often embedded within the system. Boasson (1993) explain that the software in the control systems performs actions from gathering data about the system through sensors to corrective actions and informing the

user of the current and predicted state of the system. As examples of the systems where these kinds of software are used Jaffe et al. (1991) list ships, missiles, aircrafts, manufacturing and processing plants or transportation systems. Therefore it can be concluded that industrial software is used in transportation and utilities in addition to the traditional manufacturing industries.

Koziolek et al. (2009) define industrial software applications in their conference article about the software architecture of software applications used for ABB robots as applications that control complex machines. They also have to respond to user's requests or stimuli from external sources while attaining to tight time restrictions and safety constraints. (Koziolek et al. 2009)

2.2.1. Types of industrial software

In the previously mentioned literature industrial software is closely linked to controlling a system, whether that system be an entire manufacturing plant or an individual machine. This kind of definition however seems too narrow. To get a broader perspective of what industrial software consist of an array of literature discussing aspects of industrial software was looked at.

Lichter et al. (1993) study the results of prototyping industrial software projects. They conducted five case studies to document how prototyping is used in industrial software projects and what the success factors or critical factors for failure in these projects may be. The first case study presented a configuration system for sales support in a service company. The idea behind the prototyped software was that it would help the sales as an advice system. The configurations could be made based on customer requests. (Lichter et al. 1993.) The basic idea behind this is that the software makes configurations according to external and internal restrictions case by case.

Control systems are the most common industrial software that seem to appear in literature. Boasson (1993) defines control systems in his study as systems that may operate from very simple to extremely complex environments to execute tasks with determination in order to reach a preset goal. Koziolek et al. (2009) looked into 58 robotics applications in detail they grouped these applications into five groups. One of these groups was called job control, this group of applications control job queues and the execution of these jobs. (Koziolek et al. 2009.) One of the case studies in Lichter et al. (1993) article is a process control system for machine tools. Also Kettu et al. (2008) conducted only one case study of complex industrial systems in an industrial organization, the object of their analysis being a large scale control system. Blanke et al. (1997) have not conducted a case study but their study discusses industrial automation and fault-tolerant control systems used within. They have not specified industrial software but they do emphasize the industrial environment that these systems

are used in. This is also the case with the article by Ferrarini & Carpanzano (2002) where they look into design and implementation of control and supervision systems in robotic applications. As explained earlier Jaffe et al. (1991) look into process-control in their study of software requirements analysis for real-time process-control systems. In their examples of these systems it is clear that an industrial environment is assumed for process-control software.

Koziolek et al. (2009) state that one of the industrial software categories studied in their article was job coordination. These software applications coordinate jobs on different job controllers throughout production. (Koziolek et al. 2009.) Three of the case studies from the Lichter et al. (1993) study could be categorized as coordination software. One of the case studies is a distributed information system that would coordinate information flow across different computers. The second case study that would fall into the coordination category would be the planning tool to aid the planning of heating and plumbing jobs. The third case in this category is the document management system that would help manage, revise and even produce these documents. This falls under the coordination category as the document types were in this case very similar only requiring good coordination to execute said tasks. (Lichter et al. 1993.) The title of the study by Ostrand & Weyuker (2002) is "The Distribution of Faults in a Large Industrial Software System", it is a case study of thirteen inventory tracking systems. It can therefore be construed that inventory tracking falls under industrial software. For the purpose of simplifying, inventory tracking is placed under the coordination systems. Bughin et al. (2010) bring up energy optimization and analytic software for routing logistics as examples of industrial software that can be used to move towards a sustainable world.

Koziolek et al. (2009) classify engineering and simulation systems as categories of industrial software that appeared in the software applications they studied. Engineering is explained to be software applications that enable robot programming and configuration without interfering with the production process of the robot. Simulation on the other hand gives the possibility of testing the previously engineered programs. (Koziolek et al. 2009)

Supervision systems is also one of the categories in the study by Koziolek et al. (2009). They define it as software that lets the operators control and monitor the system. (Koziolek et al. 2009.) As stated before, Ferrarini & Carpanzano (2002) also mention supervision systems, in addition to control systems, when it comes to software in an industrial setting, in their case robotics. This aspect is also touched upon by Boasson (1993) as he stated that one of the actions of control systems software is to identify and inform of the status of the system. Blanke et al. (1997) also include supervisory systems as a module of control systems.

2.2.2. Review of industrial software

Many of the literature in the previous chapter included case studies of real industrial cases of industrial software. From these literature and case examples Table 1 was constructed. As noted, six categories of industrial software were identified. The column titles embody the different categories of industrial software systems that occurred in the examined literature. The row titles represent the literature that was looked at. The cells are marked with an "x" if the industrial software system was mentioned in the respective literature.

Table 1. *Industrial software systems occurring in various literature.*

	Configuration systems	Control systems	Coordination systems	Engineering systems	Simulation system	Supervision system
Blanke et al. (1997)		x				x
Bughin et al. (2010)			x			
Ferrarini & Carpanzano (2002)		x				x
Jaffe et al. (1991)		x				
Kettu et al. (2008)		x				
Koziolek et al. (2009)		x	x	x	x	x
Lichter et al. (1993)	x	x	x			
Ostrand & Weyuker (2002)			x			

As seen in Table 1, control systems appear the most in examples of industrial software however this is not the only kind of software that can be categorized as industrial

software. In fact control systems were mentioned in six different studies. Coordination systems gathered the second most mentions, with four occurrences, while supervision systems were mentioned in three different articles. Engineering systems, simulation systems and coordination systems each gathered one mention in the studies that were observed. This table does not show the entire field of industrial software but rather works as an example of what types of software are categorized as industrial software. It serves mainly as an exemplification of the fact that not only control software should be seen as industrial software.

As stated before, these examples do not give the actual range of industrial software but rather give examples on different software that are categorized under industrial software. Perhaps industrial software has too wide of a scope to be defined in detail. Despite the fact that defining what lies within the industrial software scope is quite vaguely determined, in this thesis industrial software is defined as any software directly linked to controls, supervision and analytics of any systems within industry, utilities as well as infrastructure and transportation. The systems can be anything from fleets to single functions in individual machines.

This chapter has given an overview of what kind of software is available in the field that also this study situates in. In a way these industrial software types examine the categories of the different value propositions of the software. The targeted customer is defined by the target market of the company. In regards to the delivery models and revenue models none of these categories really examined the matter. In general, from the software industry two main business models can be identified from literature. The first being the on-premises software that is run on the customers servers and the second, Software as a Service that is run externally. Both of these business models are also associated with specific delivery models and revenue models. In addition in the industry, regarding software service solutions, a trend of software-based services has been identified. These three and their delivery and revenue models are examined in the next chapters.

2.3. On-premises software

The on-premises software can be considered as the most traditional software model. In this chapter first the delivery model of the on-premises software is examined. In the second part of this chapter the different revenue models associated with the on-premises software are observed.

2.3.1. The delivery model

The traditional software model also otherwise known as the on-premises model means that the customer for example buys a permanent license to a software application. The

software is installed, run and maintained on the customer's hardware. (Xin & Levina 2008; Waters 2005.) The maintenance is also often done by the customer's internal employees (Xin & Levina 2008). This is the way most enterprise software has been acquired for over thirty years (Waters 2005). Butler (1999) separates packaged off-the-shelf software from customized software. However, in both cases the software is run and maintained on the internal servers of the customer, so this distinction does not pose a large difference to the delivery model.

Enterprise software requires IT expertise in implementation and resources in maintenance of the application. In addition they often include hidden costs that affect the return on investment. Also security needs require a lot of expertise and perpetual upgrades by the customer. (Waters 2005.) In the traditional on-premises software new features are usually only delivered to the customer through new releases and separate upgrades. The updates that repair existing features and protections against vulnerabilities in the code are delivered in patches. Upgrades are usually sold separately and therefore acquired by the customer only when the benefits of the upgrade outweigh the cost of the upgrade. (Choudhary 2007.)

The hidden costs may come from for example delayed implementation. Also this requires ongoing administration such as managing to sustain the compatibility of different operating systems and platforms. When IT is licensed, it is often the case that companies tend to buy more capacity than they currently need but still run out of capacity rather quickly. As the software is run on the customer's hardware, clients can be faced with the snowball effect where the new software application is not compatible with the current database. And therefore, a new one can be acquired and perhaps also a new server is required all the way to having to purchase brand new hardware just to be able to support the new application. (Waters 2005.)

2.3.2. The revenue models

The most common revenue model for on-premises software is the perpetual licensing model. Konary et al. (2004) define the perpetual license as a one-time payment by the user in order to get a right to run the software as long as needed. It is important to distinguish licensing from selling, Andris (2002) explains that the licensing enables the vendor to place restrictions on how the software product is used or who uses it. The license can be determined on the number of users, computers, networks or servers (Ferrante 2006).

The high lisence fees help cover the development costs of the software within a short time period (Ojala 2013). However, for the customer the licensing model may include significant hidden costs (Waters 2005). The model might require the customer to make additional investments to hardware, installation and maintenance (Choudhary 2007).

The purchase price tends to be only a small part of the actual software costs and might therefore require budgeting decisions and large IT resources. Therefore, it could be concluded that larger firms might be more forthcoming to go for the licensing model in comparison to a small or a medium enterprises. (Ojala 2013.)

Because the license fee is usually a large investment, it causes the customer increased switching costs. This can be viewed as a good way to lock in customers from the vendors side and increase customer loyalty. (Ojala 2013.) However, at the same time the licensing model has very low predictability in revenues with its peaks and valleys in revenue streams, increasing the sensitivity to the economic climate. (Konary et al. 2004.)

The perpetual license usually does not include the right for upgrades (Konary et al. 2004). It is quite common that the vendor requires the customer to purchase technical support, upgrades and patches for a predefined time period (Ferrante 2006). This is usually achieved by selling separate maintenance agreements to the customer (Konary et al. 2004). The maintenance fee typically promises the customer corrections and bug fixes as well as upgrades and enhancements and sometimes also technical support to the software product (Butler 1999; Konary et al. 2004; Cusumano 2008). The maintenance fees are usually paid monthly or annually, the vendor's can use this to profitably lock in the customers in the long term bases. (Butler 1999.) The price is often determined as a percentage of the net or list price of the license fee. (Konary et al. 2004)

Most customers buy the maintenance agreement as a form of insurance policy to ensure a properly functioning software product rather than for the new functionalities that may come from updates. In fact, often customers do not believe they need upgrades and may refuse to pay for them separately. (Konary et al. 2004.) Butler (1999) states it wise to combine the perpetual license with the maintenance agreement as the combination also helps the customer with long range planning, since the customer will then also know the lifecycle costs of the software product. Konary et al. (2004) also note that the maintenance fees smooth out the revenue streams and therefore provide some indication of the future earnings.

According to Konary et al. (2004) while the vendors are increasingly aiming for better predictability of software revenues, there is also a shift of purchasing patterns on the customer side with a emergent emphasis on steady and recurring revenue streams. The shift on the customer side is affected by the demand for predictability in software costs. (Konary et al. 2004) The revenue model to answer to both the vendor's and customers' demands is called the subscription model, where the vendor can achieve a steady stream of earnings and the customer can disperse the payments over a long time period (Ferrante 2006; Konary et al. 2004).

Konary et al. (2004) describe the subscription model to be a repeatedly, often annually paid fee for a subscription license that permits the customer to continue the use of the software, in other words the customer does not own the license. If the customer ceases to pay the fee, the software stops working. While perpetual license dominates the marketplace, the subscription model is becoming more and more prominent. (Konary et al. 2004.) Ferrante (2006) adds that in the subscription model the customer will automatically receive the latest upgrades and newest features. The subscription model can be seen as a way to eliminate separate maintenance agreements (Cusumano 2008).

Through the subscription model, the customer also achieves greater flexibility and simplicity in contracts, which is something that according to Konary et al. (2004) the customer wants. Ferrante (2006) agrees, since the subscription model also often enables an easier possibility to try the software affordably or even for free before a more substantial commitment. However, the subscription does seem to fall short when the applications are mission critical and need constant uptime, because the subscription model does not ensure a continuum of use if for example the vendor goes out of business. (Ferrante 2006.)

The subscription model will be explained further in chapter 2.4.2. Also pay-per-use model is mentioned as a possible revenue model for on-premises software by Ojala (2013) but as it is more associated with the Software as a Service delivery model it too will be explained in chapter 2.4.2.

2.4. Software-based services

Software-based services is a phenomenon that does not appear in literature as it is. The business model is introduced in this thesis because it is a very relevant business model, which is present in the industry. In some cases it is a viable option for the other software business models introduced in this thesis. In this chapter the software-based services is explained through literature that address similar but not identical business models. These similar business models, software-enabled services and e-services, have certain traits that also define software-based services but also possess distinct differences from the business model in question. These differences and similarities are explained in the next subchapter.

2.4.1. The delivery model

In his blog-post in Huffington Post, David B. Black (2008) introduces the software-enabled services. According to Black (2008) the key aspect in software-enabled services is that the software is used as the key enabler to provide services to the customer. He introduces five aspects that define whether a business is a software-enabled services business: 1) the software is written to run the business, 2) the core business operations

cannot be conducted if the software does not work, 3) the software is only run by the service provider 4) the software is run on the servers of the service provider or a partner of the service provider, not the customer and the service provider has full control over the computing environment and 5) the main users of the software are the customers, even if some of the employees within the business might use the software as part of their job functions. (Black 2008.)

When it comes to software-based services, the list could be very similar. The software is essential in delivering the service and the software is run by the service provider. However, the main users of the software would be found within the organization. This does not exclude the option of having the customers have an interface to the software but the customer usually does not use the actual software. This is due to the fact, that in order to deliver the actual service, human expertise is often needed. An example of this would be extensive analysis based on data collected from the customers' processes. The analysis is done by the service provider based on the outputs of the software. Alternatively, recommendations or services based on the analysis done by the software are constructed and then sold to the customer as a service.

Black (2008) compares the software-enabled services to Software as a Service. According to Black (2008) the main difference between Software as a Service and software-enabled services is that with Software as a Service the software is in a sense the product and only the payment method makes it a service. In software enabled services the software is the core of the business but not the business itself. He states that the key aspect is, that in Software as a Service or on-premises software the customer buys the software to run their business, whereas in the software-enabled services the software is run in order to be able to run the service provider's business. (Black 2008.) In software-based services this is also the case. The software is critical in producing the service that is offered to the customer but the software on its own does not deliver the value.

When it comes to internal software, Black (2008) makes the distinction that internal software is not used as the primary medium of interacting with the customer. As examples Black (2008) gives online travel booking sites and online shopping. (Black 2008.) This is also the idea behind the concept of e-service. Featherman & Pavlou (2003) define e-services information systems based on software that the customer's receive via the internet. Rust & Kannan (2002) define e-services as providing service to customers over electronic networks, or to put it simply the internet. In their later work Rust & Kannan (2003) specify that in addition to the internet, electronic networks span to wireless networks and electronic environments that might include ATMs and smart card networks to name a few. Rust & Lemon (2001) proclaim that the real purpose of e-services is to provide a superior experience to the customer through interactive flow of information.

The main point behind the literature of e-service as well as software-enabled services is that the service is provided using the electronic networks as a channel. In software-based services this is not ruled out. The communication can be done through the internet in the way of remote services, remote expert consultations and so on. However, e-services in this sense are too narrow of a concept. In software based services information flow is essential to the software and service but the information flow is most important electronically from the customer side to the software, meaning for example the data from processes. The service provided from the outputs of the software can be delivered to the customer through electronic networks or in person.

2.4.2. The revenue models

In software-based services, what is sold to the customer is not software but rather a service. As the different services that are generated with the help of software can differ vastly, the revenue models must therefore also differ as immensely. The revenue models should, in these situations, be based on different service revenue models. As the thesis is focused on software solutions the service revenue models are out of scope and are not discussed in this thesis.

2.5. Software as a Service

Software as a Service is described in literature as a delivery model as well as a revenue model and often these two are very much intertwined. However, Software as a Service as a delivery model is more straight forward than as a revenue model. With Software as a Service there are more than one revenue model options for the one option of delivery model. This is why, in the next chapter, the basis of the delivery model is explained first and after that the possible revenue models are clarified. After this the different revenue models related to Software as a Service are examined.

2.5.1. The delivery model

Waters (2005) describes Software as a Service as being a new delivery method for software applications. Both Sun et al. (2008) and Dubey & Wagle (2007) term Software as a Service as a web based delivery model and Aisopos et al. (2013) define Software as a Service as offering scalable and virtual resources over the internet, as a service. The basis of the delivery model is that, unlike in licensing software, the software applications run in the suppliers' datacenters (Choudhary 2007; Ojala 2013; Waters 2005). This also means that the data of the customer, regarding the application, is stored by the vendor in the centralized datacenters (Ma 2007). This model has been around for a while, with vendors offering software applications such as emails and calendars on the online web platform to consumers, however, when vendors open the Software as a Service infrastructure to other product or manufacturing companies it is considered an

industry platform (Cusumano 2010), this is the platform that this thesis will focus on. Good examples of existing successful Software as a Service vendors are Salesforce.com and NetSuite (Choudhary 2007).

Basically the infrastructure is based on a multi-tenant architecture, which means that there is only one common code that runs in the vendor's server (Benlian & Hess 2011). The supplier of Software as a Service takes the responsibility of the maintenance of the software and servers, while the customer still has full administrative control (Waters 2005). Katzan & Dowling (2010) describe this as transferring the control from the customer to the service provider. This sets limits to the customer's customization options in regards to the main functionality and data structures of the application. On the other hand, this gives the supplier more control on the future development of the application, and the customers have to acquire all the upgrades in order to keep using the application. (Xin & Levina 2008; Benlian & Hess 2011.) Where the customer loses control over customization and functionality, Dubey & Wagle (2007) claim that opting for Software as a Service would in fact result in the customer having more control over the relationship with the vendor, because switching a vendor becomes easier with no large capital expenses committed to the software. The fact that the software is not run on the customer's premises also decreases the risk of piracy (Ojala 2013).

Waters (2005) lists the enablers of the Software as a Service business models. By enablers he means the things that have made the Software as a Service business model possible. The first enabler is the relative "homogeny and ubiquity of workstations". This means that practically everyone in business has access to the internet and everyone uses the same data communication protocols and therefore, all the customers can be reached through a single channel. The next enabler is that the physical location of data does not matter anymore and the same security methods can be used to protect the data no matter where it is physically stored. These days the applications can also exchange data with each other regardless of their geographic locations. Also the maturity of the software business has finally enabled suppliers and customers to enter into understandable and clear service agreements. (Waters 2005.)

For the vendor Software as a Service delivery model and the multitenant architecture provides certain benefits. The most important and biggest benefit of Software as a Service delivery model for the vendor is the opportunity to achieve economies of scale (Benlian & Hess 2011; Sun et al. 2008; Waters 2005). This is achieved through distributing the costs across thousands of customers (Waters 2005). Also, Software as a Service requires no or minimal client specific investments, which in turn reduces cost and ensures previously mentioned economies of scale (Xin & Levina 2008). Katzan & Dowling (2010) look at this from a tradeoff perspective. If the software is run in-house, the client has total control over the software and the processes linked to it, but the economies of scale cannot be reached. Whereas, if the software is delivered, for

example, through a cloud, the control decreases as the opportunity for economies of scale increases. (Katzan & Dowling 2010.) Waters (2005) states that Software as a Service ensures better efficiency and reliability of the functionality and uptime of the software. A study by Choudhary (2007) also suggests that the Software as a Service architecture usually encourages higher investments in quality in comparison to the on-premises software and this was seen to lead to higher profitability.

Waters (2005) also defines reliability as a benefit of Software as a Service to the customer. An efficient Software as a Service vendor can provide reliability features that individual customers could have difficulties matching internally. Often vendors promise uptime guarantees in the Software as a Service contracts that are much higher that the customers could keep up with when it comes to traditional software. (Waters 2005.) This can be related to possible quality improvements in the software applications used, that was seen as the third biggest perceived opportunity in Software as a Service adoption (Benlian & Hess 2011). Also security, data safety and disaster recovery are often better with the Software as a Service concept. Especially since the vendor can often provide these with a much lower price than what the customer could provide individually. (Waters 2005.)

The benefits of Software as a Service increase as the complexity of the mix of users increases. If the software application needs to be used by both external and internal users, the external users can easily be provided with simple secure access to the application without having to undergo corporate firewall issues. (Waters 2005.) On the other hand Xin & Levina (2008) state that when the client has a large number of users, they may be less likely to adopt Software as a Service delivery model due to possibly achieving economies of scale within the organization and therefore not benefiting from the economies of scale achieved by the vendor. The difference with these two aspect is that Waters (2005) emphasizes the heterogeneity and geographical dispersion of the users and Xin & Levina (2008) assume a homogeneity across the users and consider mere volume.

When the customers have a need for regular updates on either third-party data or to maintain compatibility, Software as a Service is often a good option, as these updates can be done automatically with no effort on the customer's side. (Waters 2005.) Foley (2004) also mentions that a benefit of the Software as a Service delivery model is the constantly up-to-date software. Likewise Ojala (2013) lists the fact that the customer always has the latest version of the software as well as the customer not having to worry about storage capacity or technical specifications of their computer, as advantages for the customer. However, if the software is directly related to the customer's core business and when the risk resulting from loss of internet connection is high, the data is better stored on internal servers. (Ojala 2013)

As stated before, security is one of the benefits of Software as a Service because the vendors can often offer better security for a significantly lower price. Nonetheless, some companies, that have high security needs, may feel that they need to have physical possession of their data due to organizational culture or policy and therefore rather steer away from acquiring Software as a Service applications. This is based on more the feeling of security rather than the actual level of security achieved. (Waters 2005.) This is also proved by Benlian & Hess (2011), they state that IT executives perceive security risks as the biggest risk factor in Software as a Service adoption followed by performance and economic risk. The security risks were related to the possibility of loopholes in the contracts that can be used to harm or exploit the customer. (Benlian & Hess 2011.)

The speed of deployment is also very fast for the customer in Software as a Service. The customer does not have to install and configure the software in order to use it. Basically Software as a Service can be implemented for the customer almost instantly. Even when there are customization or other needs, the implementation tends to be faster than with traditional software. This speed is achieved because the configurations can be done in the supplier's premises rather than sending experts to configure and customize the software on the customers' site. (Waters 2005.)

Waters (2005) states that perhaps the most compelling argument for Software as a Service is the benefit of risk mitigation. Software as a Service business model allows customers to have access to powerful software and its benefits, while at the same time protecting themselves from a lot of the risks that come with traditional software. These risks in Software as a Service are the vendor's responsibility. This way the customer also reduces the advance commitment of capital. (Waters 2005.)

Waters (2005) reminds though, that in all cases Software as a Service is not the best option for the customer. If the software application is needed by only one or few users the benefits of Software as a Service are diminished. However, when the number of users increase and the wider their geographical distribution is, the better the benefits of Software as a Service become to the customer. (Waters 2005.) Xin & Levina (2008) state that the uncertainty for service volume, not merely the volume itself, affects the tendency to adopt Software as a Service. They state that the higher the uncertainty of service volume is for the customer, the more likely they are to go for the Software as a Service model in comparison to the on-premises model. (Xin & Levina 2008.) Also the size and resources of the customer's internal IT department, have an effect on how beneficial Software as a Service is as an option. Basically the less resources the internal IT department has, the better the benefits of Software as a Service become. The capital expenditure budget as well as the operational budget of the customer also has an effect on which business model might suit them better. If there are restraints on both budgets Software as a Service tends to be the better option. (Waters 2005.)

According to Waters (2005) integration needs to other software, specifically older software, may make implementing Software as a Service difficult. Newer software applications are often engineered for much looser and easier integration than older software so Software as a Service is much better fitted when the integration needs are towards newer software applications rather than older ones. (Waters 2005.) Ma (2007) approaches this matter through unfit costs. Unfit costs mean the extra effort that customers have to face to make different software applications integrate smoothly. Modularity, use of open programming languages and loose couplings with other interfaces decrease the users' unfit costs. Overall, as the unfit costs decrease, the more the relative economic advantages of Software as a Service will increase. (Ma 2007.)

No customer is the same and this might affect the offering of the software product. The industry focus, the behavior, the culture, the regulations, the strategy or the product offering might affect the customers' needs when it comes to the software. This leads to a need for tailoring. Tailoring can be divided into two different types: customization and configuration. Customization means changes made to the source code of the software in order to tailor the product for the customer. As Software as a Service is based on a single source code for all the customers, this proves costly as the software would have to be maintained tenant per tenant. Configuration on the other hand does not require any changes to the source code and is therefore preferred by the Software as a Service vendor. The variance and tailoring is achieved through pre-defined parameters to change the functions of the application within a pre-defined scope. (Sun et al. 2008.) Also Xin & Levina (2008) agree that usually the higher the degree of required customization from the customer, the less likely the customer is to adopt Software as a Service.

Sun et al. (2008) explain that in order to define the extent of customization or configuration needed the customer requirements should be well understood. The customer segment should be identified and through this the scope of the needed variance for the application. Also, the uniqueness of the requirements should be viewed. (Sun et al. 2008.) Xin & Levina (2008) state that especially when it comes to strategic applications or applications where the business process and the software application are closely intertwined to establish competitive advantage, customization is often a requirement. Also, the uncertainty of future tailoring needs affect the customer's tendency to adopt Software as a Service, according to Xin & Levina (2008) the higher the uncertainty of future customization needs, the less likely the customer is to adopt Software as a Service as a delivery model.

2.5.2. The revenue models

With Software as a Service the customer pays for the service of using a standard software application online (Buxmann et al. 2008). Waters (2005) explains the revenue

model side of Software as a Service versus on-premises software as a rent vs. buy decision. Therefore the cost benefits are a valuable aspect in the decision. Sharpe & Nguyen (1995) state that typically companies with high cost of capital can economize on costs through leasing by spreading the cost of service over time.

According to Waters (2005) total cost of ownership is one of the most obvious points of the Software as a Service business model and the biggest benefit to the customer. In Software as a Service the total costs are contractually defined in advance, regardless of the used revenue model and compared to traditional software business model the Software as a Service model is much more transparent. In the traditional software business model the customer often faces several hidden costs such as unexpected down time, training costs and delays in implementation to mention a few. (Waters 2005) Benlian & Hess (2011) found that customers viewed cost advantages as the most sought after opportunity in Software as a Service adoption. The total costs in Software as a Service are also usually lower as the suppliers often achieve better efficiency by managing their application centrally for multiple customers (Waters 2005). Also Foley (2004) lists cost savings as one of the main benefits of Software as a Service with the customer's increased bargaining power.

According to Benlian & Hess (2011) the second biggest perceived opportunity of Software as a Service adoption is strategic flexibility, meaning the ease of changing software providers. While Foley (2004) does not directly rank the main benefits of Software as a Service to the customer he refers to the ability and ease of switching vendors as one of the main benefits of Software as a Service. This refers directly to the bigger capital investment of the traditional software licensing as well as the previously mentioned hidden costs (Waters 2005).

The subscription model is a very common revenue model for Software as a Service offering. The model is also often known as the renting model or just as Software as a Service. Most of the same benefits and pitfalls apply with the subscription model in the on-premises as well as the Software as a Service models. Ojala (2013) explains that the subscription fee offers the vendor a lot of flexibility when it comes to pricing as the price can be based on a number of different aspect from the length of the agreement to the number of users, size of the customer's organization or the functionalities of the offered software. Armbrust et al. (2010) describe the subscription model as a rental model where a negotiated subscription fee is paid to use the software for a predefined time. The customer pays for the right to use this software during this time irrespective of the fact whether the software is used (Ojala 2013).

According to Waters (2005) often with the subscription model the customer can purchase according to their capacity needs with easy immediate expansion possibilities. This leads to significant capital efficiencies, similar to just-in-time in manufacturing.

Also updates to the software can be performed regularly and seamlessly without new software installations on the customer side. These updates are performed on the supplier's side without any disturbances to the customer. (Waters 2005.)

Choudhary (2007) explains that the subscription model enables the customer to gain a small stable cash flow with no initial investment. As all the terms of costs are defined in the subscription agreement (Ojala 2013), this model presents no large sunken costs which affects the customer's willingness to pay for the service positively (Choudhary 2007). Ojala (2013) explains that as long as the vendor can uphold loyal relations with its customers, the subscription model may generate more revenues compared to alternative revenue models in the long run.

The cost of the software is understandable and predictable as the customer pays only for the use. In some cases Software as a Service is referred to as the utility model as it is a service delivery method comparable to utilities such as electricity. (Waters 2005.) The payment methods can also be described as the pay-per-use method (Greschler & Mangan 2002). Ferrante (2006) describes the utility model as a model where the customer pays per the use of the software according to how the agreement defines use. The use can be measured as per the number of times a certain aspect of the software is used, the number of transactions, the length of time that the software is running or any number of combinations from the previously mentioned units (Ojala 2013).

The pay-per-use model requires monitoring the use of the defined unit within the software and may therefore be more difficult to apply than the subscription model (Ojala 2013). Konary et al. (2004) note that sometimes even real-time monitoring may be needed. In the multiple case study by Ojala & Tyrväinen (2012) the pay-per-use model was seen as a very complex option, precisely because the model requires technical solutions to tracking usage.

According to Ojala (2013) the estimation of the future use of the software can be difficult and therefore the pay-per-use can result in unanticipated operative costs. Konary et al.(2004) similarly state the concern of unpredictable costs as a negative side of the pay-per-use model. In addition, to the vendor the revenue streams are lower and also much more unpredictable than with the earlier described subscription model. Ojala & Tyrväinen (2012) actually indicate, in their multiple case study on five software firms, that Software as a Service vendors often prefer the subscription model over the pay-per-use mode specifically because of unpredictable revenue streams. As for the customer side Ojala (2013) states that the pay-per-use model is best suited for the customers that only need the software infrequently or for a very specific purpose.

The only fairly new thing about the freemium business model is the name. Fred Wilson (2006) came up with the official definition in his blog and named the model freemium

based on the suggestion from Jarid Lukin. Wilson defined the freemium model as follows: "*Give your service away for free, possibly ad supported but maybe not, acquire a lot of customers very efficiently through word of mouth, referral networks, organic search marketing, etc, then offer premium priced value added services or an enhanced version of your service to your customer base."*(Wilson 2006.)

The freemium has since Wilson's description been an accepted term also in research. For example Teece (2010) and Cusumano (2008) both use the term in their research as possible revenue models. Teece (2010) describes the freemium as an offering of free and premium functionalities. Anderson (2009) actually states that the division can be made in four different ways. The division into free and premium can be made based on time, in a way offering a free trial for a certain time period. The division can be based on the functionalities of the software offering lower quality or less functionalities for free. Another way to divide the free and premium functionalities is by limiting the number of free users to a predefined number. And lastly the groups can be divided based on customer types where for example smaller companies get the software for free, whereas bigger companies must pay the premium. (Anderson 2009.) Teece (2010) also states that the freemium model is a prominent model for internet companies. Osterwalder & Pigneur (2010) describe the freemium as a revenue model where a small group of premium clients cover the costs of a relatively large group of free users.

The freemium model is often viewed as a marketing model where the idea is that the free offering would increase the interest in the offering (Osterwalder & Pigneur 2010). Anderson (2009) states that the freemium model reduces the risk level of trying the offered software product out before purchase. According to Shapiro & Varian (1998) the freemium model is a good way to attract attention to the offering and through the heightened interest also sell attached products. It is however important that the freemium product does not require high maintenance costs (Osterwalder & Pigneur 2010). However, Anderson (2009) states that the customer may not appreciate the value of free software and might view the offering as a scam. Murphy (2010) states in his paper, that this is the case especially in the business to business market. He states that especially when the data is sensitive to the customer's business, the customers tend to be more apprehensive and suspicious of the freemium business model. (Murphy 2010.)

2.6. Industrial software business models

Earlier the concept of business models and industrial software were explained. In addition the three software business models in regards to their delivery and revenue models were introduced. Most literature on the software business models was not set in the scope of industrial software. Nonetheless, in this thesis the three previously

presented software business models are looked at in the setting of industrial software. Figure 2 presents the scope of this thesis.

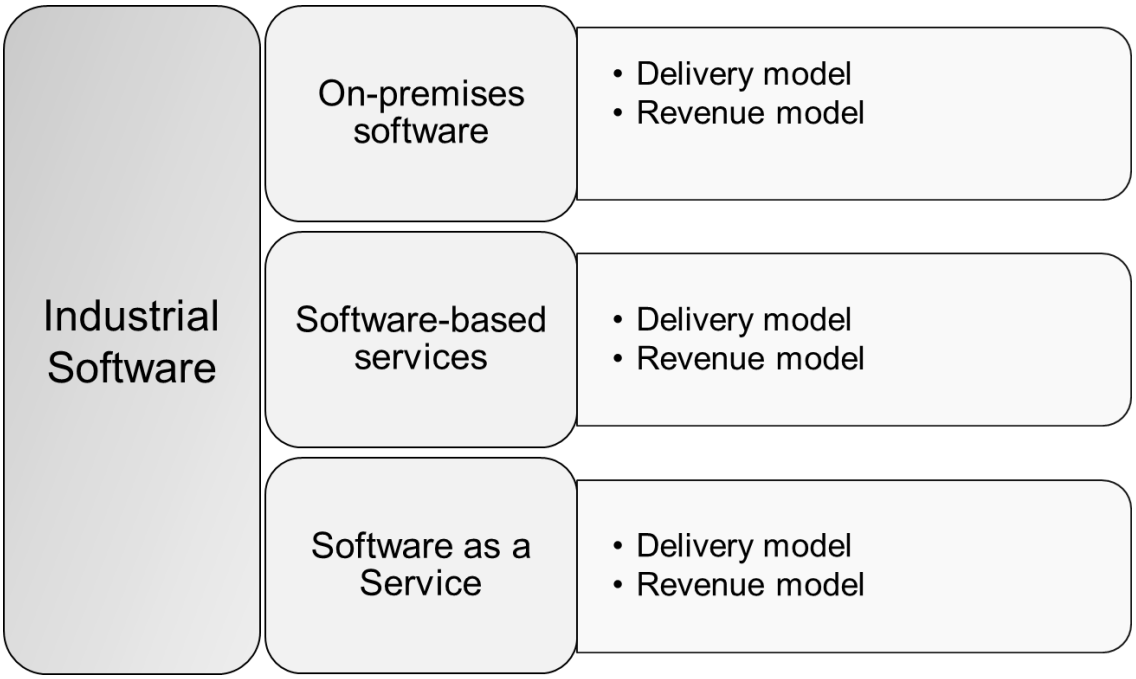


Figure 2. *The industrial software business model setting in this thesis.*

As shown in Figure 2 the different types of industrial software can be offered as on-premises software, software-based services or Software as a Service. Each of these business models also have their own delivery and revenue models that were examined in the previous chapters. Software-based services revenue model however is left out of scope as the service revenue models do not correlate with the software revenue models. The delivery and revenue models can be viewed as the building blocks that separate the different software business models from each other.

In the next chapters a summary of the different delivery and revenue models is put together. Some of the differences and similarities are examined. In the chapters only the main attributes are summarized, the models do still have other important aspects that have been explained earlier.

2.6.1. Delivery models

In Table 2 the main features of the industrial software delivery models are brought together to emphasize the main similarities and differences of the different delivery models.

Table 2. *Industrial software delivery model attributes.*

	Industrial software delivery models		
	On-premises software	Software-based services	Software as a Service
Location of software	Customer's server	Vendor's server	Vendor's server
Source code	Individual	Common	Common
Maintenance responsibility	Customer	Vendor	Vendor
Speed of deployment	Slow	Fast	Fast
Delivery channel	Customer's hardware	Service channels	Internet
User interface with customer	Yes	Sometimes	Yes
Upgrades and new releases	Bought separately	Automatic	Automatic

Table 2 summarizes the main attributes of the delivery models of on-premises software, software-based services and Software as a Service. It also point out the main differences and similarities of the three models.

The location of the software refers to where the software is run. In the case of on-premises software the software is run on the customer's servers (Xin & Levina 2008; Waters 2005). In the case of Software as a Service the software applications are run on the vendor's servers (Choudhary 2007; Ojala 2013; Waters 2005). Software-based services like software-enabled services, explained by Black (2008), are very similar to Software as a Service in this case and the software is run on the vendor side.

The source code for Software as a Service is based on one common source code according to Sun et al. (2008). The reason for this is the fact that Software as a Service is offered in a multitenant architecture (Benlian & Hess 2011; Sun et al. 2008; Waters 2005). The same principle applies for the software-based services. As the on-premises software is offered separately to each customer each customer and the software is run on the premises of the customer, each customer has their own software and therefore each software has its own individual source code.

Xin & Levina (2008) claim that the maintenance of the on-premises software is the responsibility of the customer's internal employees. The supplier of Software as a Service on the other hand takes the responsibility of the maintenance of the software (Waters 2005). As stated before, in the case of software-based services the software runs on the service provider's servers and is therefore the responsibility of the vendor to maintain the software. Also, similarly to the software-enabled services, as Black (2008)

implies ensuring that the software is running is essential as the software is an essential part of running the business and creating the service.

Waters (2005) claims that the deployment of SaaS is very fast as no installations or configurations are needed. On-premises software conversely often encounters delayed implementations as the installations need to be done on the customer site. (Waters 2005.) Software-based services do not need any installations on the customer site similarly to the Software as a Service model.

In the case of on-premises software the software runs on the customer's hardware (Xin & Levina 2008; Waters 2005). It can therefore be stated that the delivery channel is the customer's hardware and how ever the software has been installed on the customer's servers. Software-based services offer customers actual services so all different service channels are possible. Software as a Service then again is defined as a web based delivery model as it is delivered utilizing the internet (Aisopos et al. 2013; Dubey & Wagle 2007; Sun et al. 2008).

Regardless of where the software is run in the case of Software as a Service and on-premises software the customer is the main user of the framework. Unlike software-enabled services explained by Black (2008) the software based-services do not rule out, but similarly do not require a customer interface.

For on-premises software new features are usually only delivered to the customer through new releases and upgrades that need to be bought separately (Choudhary 2007; Konary et al. 2004). With Software as a Service, on the other hand, the upgrades are installed automatically and the upgrades are mandatory (Xin & Levina 2008). Software-based services act similarly to Software as a Service in this case because of the similar location of the software.

2.6.2. Revenue models

Table 3 summarizes the attributes of the industrial software revenue models explained earlier. The table is constructed from the literature reviewed in the previous chapters. In the table the main differences and similarities are evident.

Table 3. *The industrial software revenue model attributes.*

	Industrial software revenue models				
	Perpetual license	Maintenance agreements	Subscription	Pay-per-use	Freemium
Delivery model	On-premises software	On-premises software	On-premises software and Software as a Service	Software as a Service	Software as a Service
Revenue flow	Fluctuating	Steady	Steady	Fluctuating	-
Predictability of costs to the customer	Unpredictable	Predictable	Predictable	Unpredictable	Predictable
Switching costs	High	Low	Low	Low	Low
Correlation to utilization rate	Low	Low	Low	High	Moderate
Initial investment	High	Low	Low	Low	Low

The first row on Table 3 is the delivery model, thus connecting the delivery model and revenue model parts of the business model together. As explained before from literature on-premises software is associated mainly with the perpetual license revenue model but also to the maintenance agreement often sold in addition to the perpetual software and the subscription model (Konary et al. 2004). The subscription model is also the most common revenue model for Software as a Service (Ojala 2013). In addition to this also pay-per-use (Ferrante 2006; Ojala 2013; Waters 2005) and freemium models (Morris et al. 2005) are used as revenue models with SaaS.

Konary et al. (2004) explain that the perpetual lisencing model causes fluctuation in the revenue streams because of the one-off payments made only when the software is initially acquired. Maintenance fees are then used to smoothen out the revenue streams as the maintenance fees are paid in regular intervals, usually annually or monthly. (Konary et al. 2004.) Choudhary (2007) states that one of the benefits of the subscription model is in fact the stable cash flow it generates. The pay-per-use method is as Waters (2005) explains a revenue method were payments are made based on use.

Konary et al. (2004) note that as the use can be unpredictable so can the revenue streams of the vendor. For the freemium model the revenue flow depends on the way that the premium functionalities are paid for so the fluctuation or steadines of the revenue streams cannot be defined.

Even though the price of the perpetual license is predetermined in the contract the on-premise software (Ferrante 2006), licensing often includes hidden costs as Waters (2005) and Choudhary (2007) state including for example later maintenance and upgrade costs. The cost associated with the maintenance agreement can be determined predictable as it is defined in the contract for a predefined time for predefined amounts to be paid within steady intervals (Konary et al. 2004; Butler 1999). Konary et al. (2004) imply that the subscription model solves the problem of unpredictable costs by dividing the cost over a longer time period and eliminating extra maintenance costs. The pay-per-use model can according to Ojala (2013) result in unpredictable operational costs as the need to use the software can often be very unpredictable. The freemium model on the other hand often determines the payments, and hence the costs, in the premium contract.

Ojala (2013) associates the perpetual license model with high switching costs due to the large initial investment. In the case of Software as a Service, Foley (2004) and Waters (2005) state that switching vendors is fairly easy. This is especially in regards to the subscription model but as the other revenue models regardless of being on-premises or Software as a Service models lack a large investment the switching costs can be assumed to be low.

The pay-per-use model is the only revenue model that directly associates the actual use of the software and the payment method together. For this reason in the pay-per-use model sufficient monitoring is needed (Ojala 2013; Konary et al. 2004). The Freemium model can be assumed to having moderate correlation with the usage. This is based on the assumption that the customer pays for the software only if they use the software and sees value in the software.

As stated before only the perpetual license model includes a high initial investment when beginning to utilize the revenue model (Ojala 2013). The other revenue models either have small reoccurring payments or smaller payments based on the utilization rate but do not require any initial investments to start using the software.

3. RESEARCH METHOD

This chapter explains the research methods that were used in this thesis. The reasons for the specific methodologies and techniques are explained. First the research strategy is described. Following with the ways of data collection and finally explaining how the data was analyzed.

3.1. Research strategy

The research in this thesis was conducted as a multi-method interpretive research. Hennink et al. (2011, p. 9) describe the interpretive approach to being one of the main approaches in qualitative research. Fisher (2010, pp. 58-60) outlines that the interpretive research builds on the researcher to form structures out of interpretations rather than determined variables. The idea is that the researcher attempts to map out the complexity of the views of people on the topic at hand. (Fisher 2010, pp. 58-60.) Hennink et al. (2011, pp. 14-15) explain that in the interpretive research the research topic is looked at through understanding the experience the people have in the subjective perspective of those people. The subjective view leads to the idea of a reality with multiple perspectives. (Hennink et al. 2011, pp. 14-15.) In the interpretive method people develop their ideas through debate and conversation that is conducted by the person themselves and with other people (Fisher 2010, p. 60).

For this thesis the interpretive approach was used because the organization under examination was not at the stage where the topic could be observed first hand through a case study. All the business models were not yet in use within the organization so the topic had to be looked at through the experiences and knowledge related to the subject and the people's views of the future.

The point of multiple perspectives of reality was taken into consideration and therefore the people approached were selected to represent a wide range of people with different positions and perspectives on the research topic. Another way to minimize the subjectivity of the results in this thesis was to use a multi-method approach. Saunders et al. (2009, p.152) define a multi-method approach as the type of research where the data is collected using different data collection techniques related to the same methodology. In this case a multi-method qualitative study is conducted. Tashakkori & Teddlie (2010) explain that using multiple methods allows the researchers to better trust the results. Mason (2006) states that in the situation, where the reality is multidimensional, due to it

being built on the social experiences of people, the understanding on the situation may be insufficient if the phenomenon is viewed only from one dimension.

In this thesis the multi-method study includes three different data collection techniques. The techniques used are the focus group discussions, individual interviews and a workshop. These techniques are explained further in the next chapter.

3.2. Methods for data collection

This chapter is divided into the three different data collection techniques used. First the focus group discussions are examined in the initial data collection chapter. After this the supplementary unconstructed interviews are looked at following a review into the workshop technique used for the validation of the framework with a weak market test.

3.2.1. Initial data collection

The first round of interviews was conducted as unstructured focus group discussions with people within the organization that have a wide understanding of their own field as well as the organization as a whole. Krueger & Casey (2014, p. 2) explain that in focus groups the participants are specifically selected because of their particular common characteristics that relate to the topic of the focus group. In the case of this thesis, these specific characteristics were the participants' extensive knowledge of the software business as well as the power and automation industry. Each participant had a specific knowhow on their own business unit or other function within their company but also about research and development, software and the organization as a whole. The idea was to establish the significant features of the organization that would impact the success factors, the benefits or perhaps the pitfalls of the different business models and their hybrids. The aim of the conversations was to identify what factors serve as decision points on which way business models would suit different situations.

It was requested that the participants remain anonymous in this thesis. And while all specific titles cannot be disclosed, it can be stated that the participants were selected from various positions on various levels of the organization. A couple of the participants were researchers from the ABB corporate research, some held high managerial positions in countries and local as well as global business units. Also the global strategy team and software product groups were represented from a managerial level. Overall, the participants represent a good sample of ABB with knowledge from the product business, software business, service, technology and strategy.

Krueger & Casey (2014, p. 2) state that in the focus group discussions the researcher creates a nonjudgmental environment for the participants where they can share their ideas and perceptions without having to reach any consensus on the topic. They also remind that the researcher should organize more than one focus group discussion. Each

group should have about five to ten participants and be led by a skilled interviewer or moderator. (Krueger & Casey 2014, p. 2.) Silverman (2013, p. 213) explains that informal discussion around a specific topic is usually encouraged in the focus group discussion sessions.

In the case of this thesis five focus groups discussions were organized. The focus groups consisted of about two to five participants each. All in all the focus groups had 17 participants. Bigger groups were not possible due to scheduling issues and the fact that the focus groups were organized in three different countries: Switzerland, Finland and Germany. This way a wider view from the organization could be achieved.

The interviews were scheduled to be 1.5 to 2 hours long. Silverman (2013, p. 213) describe that in focus group discussions some stimulus materials are often provided. Therefore, a short description of three business models based on literature (appendix 1) and a sheet of discussion topics (appendix 2) were handed out to raise ideas on the topic. The short review of the literature on the three business models and a discussion topics list was provided to the participants a few days before the scheduled interview. The same information was also handed to the participants in the beginning of the session. After this a round of introductions was conducted and a short description to the aim of the thesis explained. After this the floor was open for anyone to give their first impressions on the material given to them.

The situation on its own was kept as a free conversation where the participants could feed off of each other's ideas. Stewart & Shamdasani (2014) actually state that one of the advantages of focus groups is that the group members can build on and react to each other's ideas. This is an especially big advantage in the case of this thesis as the topic of discussion has fairly little research in the industrial setting, allowing the data from the groups to be richer.

Silverman (2013, p. 213) says in his book that typically the discussions are recorded and transcribed for further analysis. The discussions also in this case were recorded conversations between the focus group members. The interviewer's participation was only needed when clarifications were required or it was fruitful to take a certain analogy or idea further. Questions were asked to spark new ideas when the conversation seemed to die down.

3.2.2. Supplementary data collection

A supplementary set of interviews were conducted. The supplementary interviews were conducted as unstructured interviews. Klenke (2008, pp. 125-126) explains that unstructured interviews are interviews, where no formal interview questions or structures are defined. The questions asked during these interviews are often open

ended. One of the advantages according to Klenke (2008, pp. 125-126) is that more in-depth information can be acquired. (Klenke 2008, pp. 125-126.)

The supplementary interviews were conducted to perfect and better the already constructed framework that was based on the initial data collection. Due to this the unstructured interview seemed a good fit as more in-depth analysis was expected from the interviewees on the framework and free critique was encouraged. The participants were free to stop and ask questions or challenge the steps or parts of the frameworks at any time during the call. Also open questions were asked to initiate thoughts on different parts of the frameworks.

The participants in the supplementary interviews were all experts in the field of software with experience coming from research and development or the actual software offering from their respective business unit. All of the interviewees held higher managerial positions.

Klenke (2008, p. 126) states that one of the disadvantages of the unstructured interviews is that they are very time consuming and expensive and therefore only very small samples are usually feasible. This is why only four key people were interviewed for the thesis over both the phone and in person, the interviews lasted approximately an hour. The participants concluded from people from both closer to operations as well as management. The interviews were not recorded, but notes were taken during the interview

3.2.3. Framework validation

In order to validate the framework a weak market test was performed by presenting the final framework to the Group Service team, for on behalf of which the thesis was conducted, and the participants from the initial focus groups and supplementary interviews. Hakkarainen (2006, p. 160) defines the weak market test to be passed when the examined construction is either in use in the market or someone wants to use it. This was deemed as the most viable option for framework validation because of the limits in time and resources.

The validation was done through two workshops. According to Brooks-Harris & Stock-Ward (1999, pp. 6-8) a workshop can be a short-term learning experience in which active learning is encouraged. The workshop includes a facilitator, who can encourage the learning amongst the participants of the workshop. (Brooks-Harris & Stock-Ward 1999.) The workshop was chosen as the forum to implement the weak market test because of the possibility to engage a larger number of people in a single session. Also then the education, in this case presenting the final framework and explaining the basis of the research, can be followed by an open conversation amongst the participants on the topic encouraging active learning. The results from the open conversation stated the

premise for the weak market test. If it was mutually agreed that the framework is viable for use with or without minor changes the weak market test was deemed successful and the framework validated with required changes.

The first workshop was held for the team on behalf of whom the thesis was conducted. There were eight participants for this workshop. This would also be the team that decides on the use of the results from the thesis. This workshop validates the usability of the results. The other workshop was conducted to the participants from the focus groups and the supplementary interviews. There were thirteen participants present for this workshop. This workshop validated the content of the results.

3.3. Analysis of data

The recordings from the focus group discussions were transcribed. For the scope of this thesis it was deemed that only theme based transcription was necessary. The main reason for this was that the focus groups discussions were facilitated to bring out specific issues about the topic in a very specific setting. In Corbin & Strauss (2008, p. 66) state coding to be the activity of bringing the raw data to a concept level. In this case the coding and transcribing were done simultaneously. Key words and phrases from the interviews were transcribed. These key words and phrases represented the main topics that were talked about during the discussions.

By examining the transcriptions, commonalities among the six different interviews, were detected. The commonalities were clustered and the clusters were each given a title. When combining with the knowledge from the literature review the titles were formed as questions. Saunders et al. (2009, p. 492) define this as categorizing. These categories were once more crosschecked with literature gathered about the alternative business models. After this the categories were prioritized by making connections and establishing an order of importance to the differentiating factors that the categories presented.

The notes from the supplementary interviews were reviewed against the thus far existing framework. If the comments sufficiently indicated that changes needed to be made the framework was refined accordingly. If the views of the interviewees differed, these answers were prioritized based on their knowledge on the subject at hand. The notes from the validation processes were reviewed in regards to the initial framework and changes were made accordingly.

4. RESULTS

The analysis of the focus group discussions with crosschecking with literature resulted in a decision framework of two separate decision trees one for the value delivery model and the other for the revenue model to be chosen. The framework is mainly built on yes or no questions. The framework was built in two stages. The initial framework was built on literature and the information from the focus group discussions. The framework was then perfected through supplementary interviews with key people within the case company. And finally, the framework was validated with two workshops held with the participants of the previous steps and the team that the thesis was conducted for.

The results are demonstrated through the framework, built from the results of the interviews as well as the literature review, as they both played an equal role in building the framework. Later, in the discussion part of this thesis the framework as a whole will be analyzed and the use of the framework discussed.

4.1. Delivery model

The initial framework is a decision model that should be gone through with a possible software product or service in order to determine which delivery model would work best and deliver the best possible value in each case. The second decision model determines the best revenue model to be used based on the targeted customer and delivery model, but this part will be explained in chapter 4.2.

The decision tree for the delivery model consists of questions that can be categorized in four subgroups. The first six questions can be seen as obstacles that determine whether the external servers can be utilized at all in the delivery of the product. Some of these obstacle questions have sub questions that determine if the obstacle can be bypassed. The second subgroup are the opportunities with both soft and strict criteria, these will be explained later. After this two restrictions are introduced, they do not bring any additional value to any of the delivery models but may affect the choices made, and they also may come to play if the opportunities are not seized. The fourth subgroup can be looked as the final delivery models with a few defining questions that do not relate to the delivery vehicles but directly to the offered product. The databases: customer's internal server and external server, are connection point that separate the other subgroups from the final fourth subgroup. The databases act as the delivery vehicles. The subgroups and their relations to each other can be seen in Figure 3.

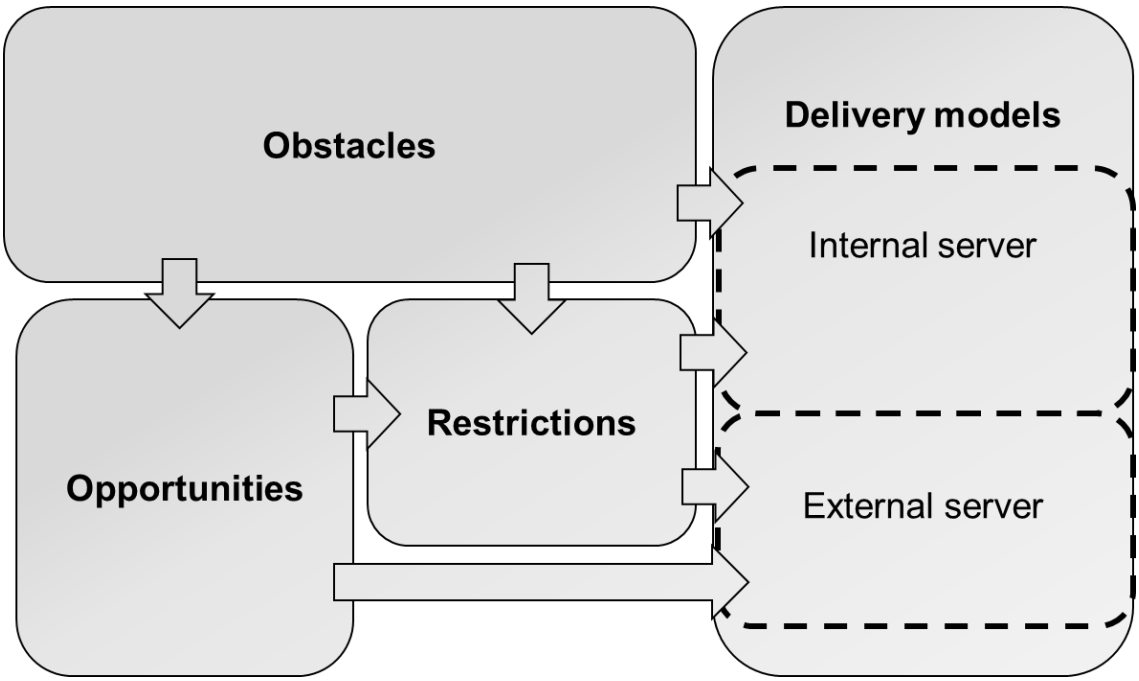


Figure 3. *The delivery model decision tree subgroups and their relations to each other.*

The boxes in the Figure 3 represent the subgroups and each arrow represents the possible outcomes each subgroup may have, pointing out which subgroup may have to be considered next. It must also be noted that the delivery models subgroup is divided based on the delivery vehicles. This was done to emphasize the fact that only certain routes can lead to the internal and external server based delivery models. These routes differ not only on the detailed level but also on the larger scale.

4.1.1. Obstacles

Figure 4 shows the first subgroup of the decision tree for the delivery model. The chart is read primarily from top to bottom following the arrows according to the answer of the question. The questions that are placed on top of each other on the very left of the figure are the main obstacle questions and the question boxes more to the right are the sub questions that look for the option of bypassing said obstacle

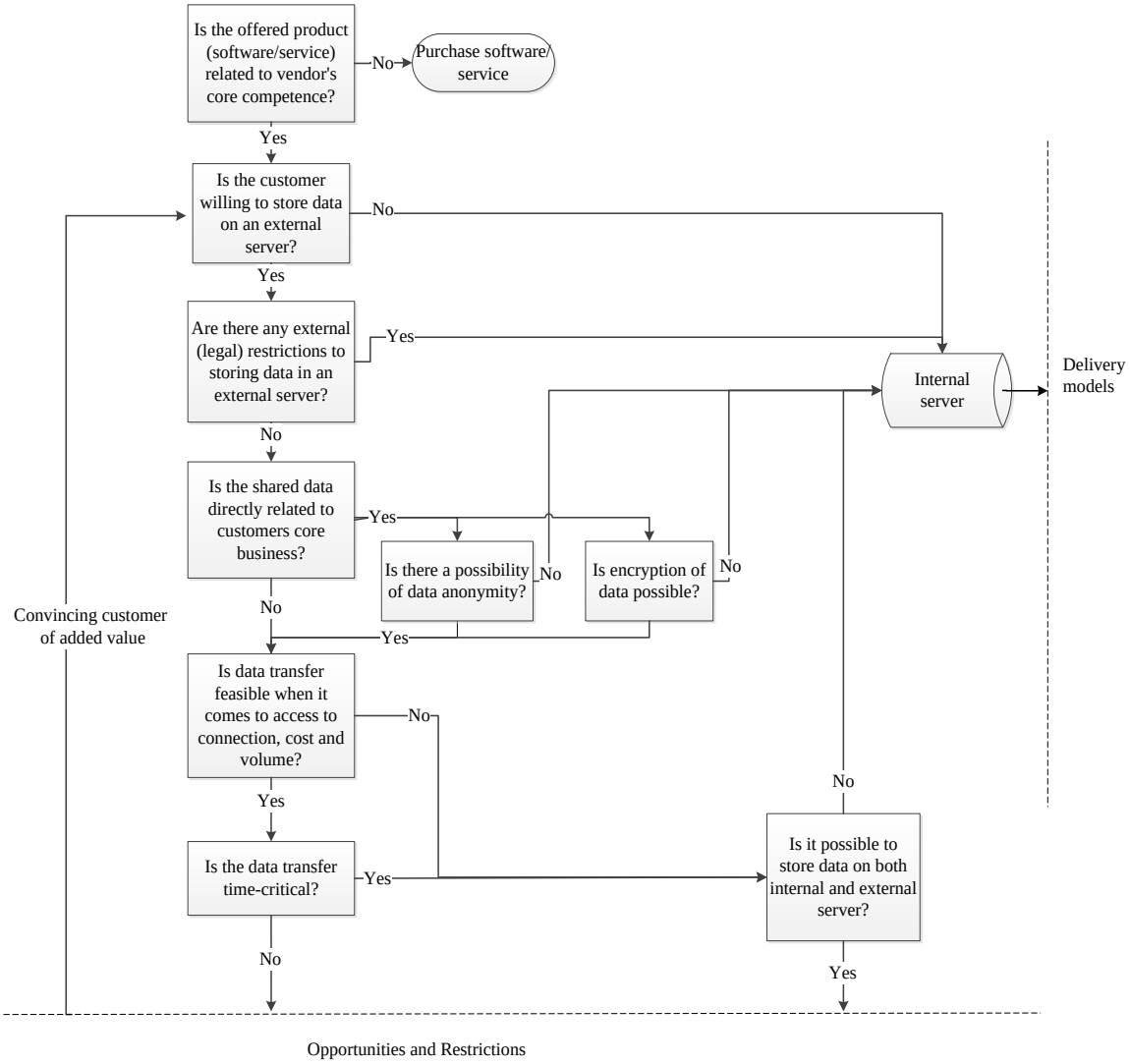


Figure 4. *Obstacles subgroup of the decision model portrays the questions that stand in the way of using an external server.*

As Shafer et al. (2005) state in the business model the value creation should happen through the firm's core competences. Also Morris et al. (2005) refer to the internal source of advantage as a key component of a successful business model. This came up in the interviews as a key starting point in offering a new product. In order to offer a product or service, it needs to be directly related to the core competences of the case company. This seemed especially important as the case company is not a software company but rather an engineering company with a long history as a pure product company with the core competencies in industry know-how and its industrial products. As stated in the first interview:

> *"The offered software product isn't good because we know software but rather because we know our products and processes."*

Therefore, it was concluded from the interviews that the first question in the framework should be the question of core competence.

Is the offered product related to vendor's core business? If this is not the case the software or service should be purchased elsewhere and the decision process ends. Obviously this works under the assumption that the product is deemed profitable and otherwise a good fit for the company portfolio. If the answer to this question is yes the decision model moves to the next question.

The customer is a key element in any business model, this is confirmed in the business model literature by many researchers such as Tsvetkova & Gustafsson (2012), Morris et al. (2005), Shafer et al. (2005) and Chesbrough & Rosenbloom (2002) to name a few. In all the focus group interviews it was stated that the offered delivery model should be dependent on the customer needs and characteristics. It was stated that the willingness to share data is highly dependent on the customer or industry characteristics. These may be the size of the company, the forward-looking culture of the company or industry or how educated the customer is on the safety of sharing data. The utilities industry was brought up as an example of an industry with very traditional values, resulting in lower tendency to adopt new technological innovations. While in literature the size is often linked to the revenue model for example Xin & Levina (2008) hypothesize that the customers with larger user volumes would be less likely to adopt software as a service. The education level of the customer is supported by Waters (2005), who states that the security issues with SaaS are often only issues of customer's perception of data security.

Is the customer willing to store data on an external server? The customer cannot be forced to adopt the delivery model of Software as a Service and if the product is profitable to be offered to the targeted customer who is not willing to store its data externally whether it be for security or other reasons the software product should be delivered using customer's internal servers. There is a feedback loop to this question from the opportunities offered by the delivery models based on external servers, but this will be explained later. If the customer is willing to entertain the idea of storing data on external servers the decision model leads to the next question.

Outside of the customer's willingness there may be other external restrictions to the data storage. In the interviews it came up that publicly listed companies for example may not be allowed to store data for example related to finances on servers owned by a third party. Schnjakin et al. (2010) in their article about cloud computing give the European Union's Data Protection Directive as an example of a legal restriction that restricts the movement of specific types of data as well as the processing and the access to this data. According to Smedinghoff (2008, p. 2) laws and regulations concerning especially corporate governance, financial information, individual privacy and sensitive business

data among others are designed to make sure that companies take care of their data security sufficiently.

Are there any external (legal) restrictions to storing data in an external server? If there are binding external restrictions for data collection to external servers the data has to be stored on the customer's internal servers. If there are no external restrictions the framework leads to the next obstacle question.

In the every focus group discussion it was stated that data that is directly related to the customer's core business is a very significant obstacle for data collection to an external server. In the third focus group interview it was stated that:

> "*In for example the utilities industry the production process is the source of the competitive advantage and is therefore too sensitive for the customer to leave the site.*"

The thought of possible data anonymity or encryption was entertained thus also enabling interesting new business models such as anonymous benchmarking services.

Is the data directly related to customer's core business? If this is not the case, there should be no problem in moving to the next obstacle question. However, if the data is related to the customer's core business the framework leads to two sub questions: *Is there a possibility for data anonymity?* And: *Is encryption of data possible?* If the answer for either of these questions is answered with a yes the framework leads to the next obstacle question. If the answer is no the data should be stored in the customer's internal servers.

The foundation of Software as a Service is that it is delivered online to the customer as stated by Sun et al. (2008) and Dubey & Wagle (2007) and operated on the vendor's servers as stated by Choudhary (2007), Ojala (2013) and Waters (2005). It can consequently be deduced that the collection of data also usually requires an internet connection. Especially in the second focus group interview it was ephasized that in some industries the working environment may restrict the possibility of data tranfer. The amount of data, the speed of the data connection, the existance of the connection and the price of the data transfer are all in connection to how feasible it is to store the data on an external server. However in the same focus group interview it was stated that this restriction could be bypassed, if the data could also be transferred to the external servers in addition to the internal ones. In this case it would require that there would have to occur intervals when data transfer would be feasible and cost effective.

Is data transfer feasible when it comes to access to connection, cost and volume? If the data transfer is feasible the next obstacle question can be asked. If the feasibility is not reached the framework leads to a sub question of: *Is it possible to store data on both*

internal and external server? If this is not possible the data should be stored only in the internal servers. However, if the double collection is feasible the framework leads to look into the opportunities subgroup and the restrictions subgroup simultaneously.

Three of the focus group interviews it was pointed out that the more time- and mission-critical the data is to the customer's processes the less likely the customer is to allow the external collection of data. In the literature for example Weinhardt et al. (2009) and Weil (2007) see that in the future more mission-critical needs can be met, but they refer systems such as Enterprise Resource Planning and do not take into account for example the production processes of the customer. In the fifth focus group interview it was declared that:

> "*The delivery model depends on the intervals of collecting data. When the reaction time is short the software is often embedde in the hardware product but the longer the needed interval of data collection, the easier the remote data collection and software is to implement.*"

Again the time-criticality could be bypassed through double collection. This could enable such services related to Software as a Service as preventive tracking through trend monitoring.

Is the data transfer time-critical? If so *is it possible to store data on both internal and external server?* The subquestion is the same as explained before with the same endings for both answers. If the transfer is not time critical thoug the framework moves to examine the opportunities and restrictions of the external server.

4.1.2. Opportunities

Figure 5 shows the opportunities of the external server based delivery model. The opportunities are divided into two: the soft criteria and the strict criteria. The division is made because all the opportunities do not pose same importance in the stated choises. The soft criteria include opportunities, that in order to be siezed, do not rule out storing data in the customer's internal servers but the opportunities can be exploited easier or more cost effectively when storing the data on the vendor's servers. The strict criteria on the other hand are opportunities that require the use of external servers in order to gain advantage.

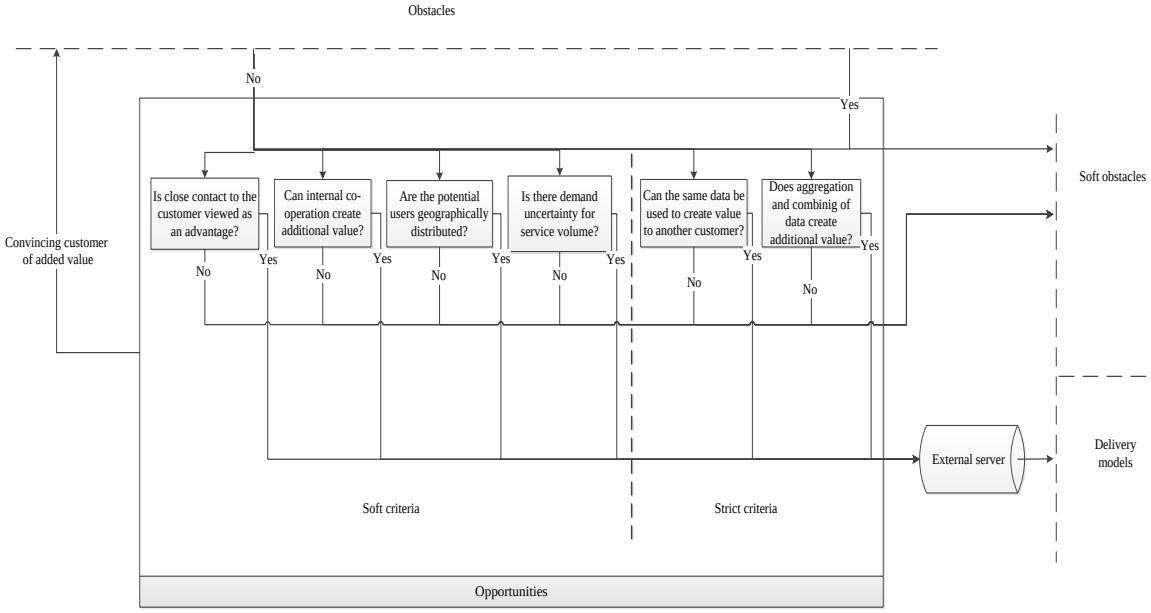


Figure 5. *Opportunities of the external server based delivery model.*

Figure 5 will be explained in order moving from left to right. The opportunities are not in any paticular order. They do not rule any of the other opportunities out and they should all be concidered equally when going through the framework.

On the very right of Figure 5 the previously mentioned feedback loop can be found leading back to the second question in the obstacles subgroup. This feedback loop is connected to all the opportunities and indicates that if any number of these opportunities could be seized the knowledge should be used to persuade the customer to change their view on willing to store data on the external server.

Ojala (2013) and Butler (1999) relate the on-premises software to customer lock-in through the pricing methods commonly used with the on-premises delivery model. Ojala (2013) states the lock-in and customer loyalty to be formed through the higher monetary investment to the software through the licensing fee. Butler (1999) on the other hand connects the lock-in to the maintenance fee agreements. The industry experts in the focus group interviews on the other hand stated that the Software as a Service business model would automatically create a closer connection to the customer and through this establishing a greater lock-in effect. While the literature on the lock-in is contradicting, Waters (2005) states that the ease of switching vendors is a very important benefit for the customer, when adopting Software as a Service. Thus can be concluded that the lock-in in Software as a Service needs to be created through closer contact to the customer. In the focus group interviews it was viewed that through using an external server and continuous online connection to the customer communication with the customer is easier to execute with the Software as a Service model. Especially the ease and speed of getting feedback on the software, from the customer, was accentuated in two of the focus group interviews. Choudhary's (2007) statement of

Software as a Service leading into higher investment in quality and thus profitability would enhance the benefit of getting direct and immediate feedback from the customer.

Is close contact to the customer viewed as an advantage? This question is viewed in the framework as a soft criteria within the opportunities. This is because close contact to the customer, and the opportunities that stem from within the connection, can also be achieved by storing the data on customer's own servers. Nevertheless, this opportunity is siezed easier through the external servers as then the vendor has an automatic online connection with the customer. If this opportunity is not valid, the framework leads to a question about the customer's IT resources in the restrictions subgroup, which will be explained later. If this is a valid opportunity, the data should be stored in the external server database.

Ease of internal cooperation was emphasized as a great opportunity for Software as a Service in all but one focus group interview. This was based on the assumpiton that less data would be lost in the customer's servers, thus increasing the ability tro learn from other solutions offered by the case company. In one of the interviews it was indicated that:

> "*With the data being stored in the internal servers it becomes a great opportunity to find the reoccuring elements in the different customer solutions*".

This idea brings the ooportunity to develop and better the solutions more effectively. The discussion of internal cooperation brought to surface ideas of unifying portal based solutions that would effectively unify the case company.

Can internal cooperation create additional value? Again this opportunity can be reached also through the on-premises model but as the data stored in the vendor's servers is easier accessible to the vendor and can therefore be more effectively utilized for cooperative efforts internally. Therefore, if this opportunity is valid the data should be stored in the external servers and if not the customer's IT resources come into play as before.

As Morris et al. (2005) state the geographical dispersion of customers can make a difference in the business model of a business. This is especially the case when comparing the Software as a Service and the on-premises model. Unlike before the physical location of data does not affect the way applications communicate and function, this is one of the main enablers for the entire Software as a Service business model according to Waters (2005). On the word of Xin & Levina (2008) as the number of users increases, the less likely the customer is to adopt the Software as a Service model. However Waters (2005) reminds that regardless of the sheer volume of the users if the geographical distribution of the users increases the more beneficial Software as a

Service becomes to the customer. Waters (2005) also implies that in general the heterogeneity of the users increase the benefits of Software as a Service.

Are the potential users geographically distributed? This opportunity is situated within the soft criteria as the on-premises model does not exclude heterogenity of the users but the complications related to a widely distributed userbase or otherwise varying user base can be solved more cost effectively when the data is stored in the external servers. Thus the answer yes leads to the data storage in the external servers and the no leads to the question of customer's IT resources.

Xin & Levina (2008) as well as Ma (2007) state that demand volatility increases the adoption of Software as a Service becomes more fruitful to the customer. When the software runs on the external servers, depending on the used revenue model, it is very easy to increase or decrease the use more cost effectively. This opportunity is also looked into in more detail in the revenue model decision chart.

Is there demand uncertainty for service volume? Based on literature the volatility of the service volume brings vast opportunities in adopting Software as a Service. Demand uncertainty does not rule out the on-premises model but it may make the customer more reluctant to enter into an on-premises agreement with the vendor because increasing and decreasing usage cost effectively becomes more of a hinderance. Therefore, if this opportunity is present, the data is encouraged to be stored at the external servers and if not, the question of customer's IT resources comes to play.

In all of the focus group interviews the potential to utilize the externally stored data to serve a party other than the initially targeted customer. It was stated in one of the focus group discussions that:

> "*There lies a potentiel to learn about our products, that can be used as an input to R&D as well as portfolio development.*"

This triggers the idea of an internal customer as a secondary party to create value to. The idea of an external secondary customer was also not ruled out but it was stated that the offered value to the secondary customer should not be in contradiction with the needs and requirements of the primary customer.

Can the sama data be used to create value to another customer? This opportunity is situated within the strict criteria as the on-premises model does not enable this opportunity. The opportunity can only be reached by storing the data on the vendor's servers. Therefore if this is an opportunity that can be reached with the selected product the data should be stored on externeal servers, if not the customer's IT resources should be looked into.

The most comments and ideas regarding the opportunities in the focus group interviews were in connection to the aggregation of data and collecting data in the long term to improve processes as well as combining data from different sources. It was seen that as the amount of data over time increases, the more value the data would gain if correct analytics were implemented. This data would not have to be from the same target customer or even industry but rather, it was seen that having access to various kinds of data over time from differet customers, the value and quality of the offering could be increased by combining and aggregating said data.

*Does aggregation and combining of data create additional value?*As the combining and aggregation of different customer's data is not possible if the data stays in the customer site this opportunity is within the strict criteria. Therefore if this opportunity is a fruitful option the data should be stored on external servers and if not the question of customer IT resources should be considered.

4.1.3. Restrictions

Figure 6 shows the next subgroup of the delivery model part of the framework. In the restrictions subgroup the two questions depict scenarios where the customer might opt for the on-premises model despite some opportunities. These restrictions need to be considered simultaneously with the opportunities. They do not, however, offer any additional value to any of the delivery models but rather state some restrictions that need to be considered when choosing the delivery model. They also do not make the opportunities redundant if they lead to a different end result than some of the opportunities might lead to. The valuation of the opportunities and restrictions need to be done case by case.

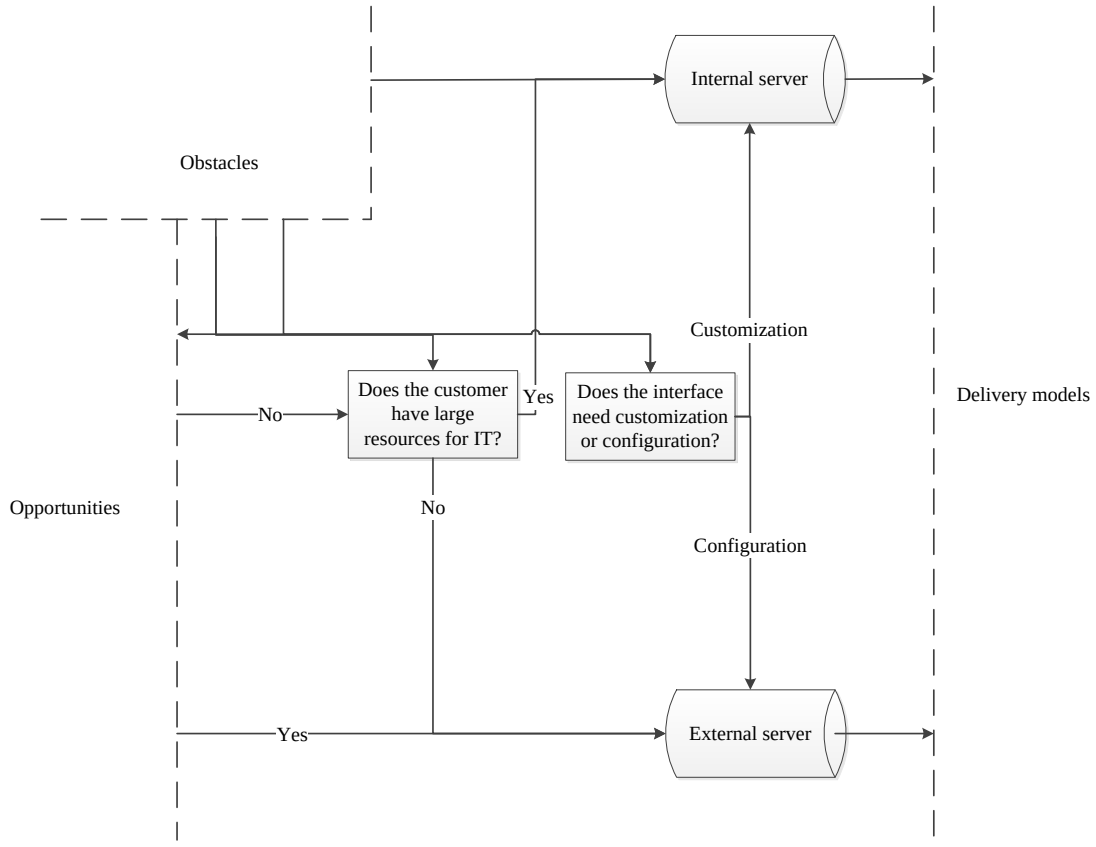


Figure 6. *Restrictions depict the questions that may effect the customers choise to opt for the storing the data on the external server.*

Both Waters (2005) and Ojala (2013) associate customers with large IT resources to the on-premises model. Waters (2005) indicates that when the budget for IT resources is low the customer is more likely to go for Software as a Service as the vendor is then responsible for maintenance, upgrades and functionality. Also Ojala (2013) talks about budgets stating that as the on-premises software includes more than just the price of the software, the on-premises model often requires larger IT budgets. Xin & Levina (2008) hypothesize that customers that have larger IT capabilities internally would be less likely to adopt the Software as a Service model and would rather go for the on-premises model. Also, in the interviews the customer's resources in IT was seen as something that would affect the customer's decision on which delivery model to choose. However, the resources were not seen as a deal breaker and it was stated that the Software as a Service specific opportunities would often outweigh the fact that the customer has a previous investment in IT, which the customer could not utilize with a given software service solution.

Does the customer have large resources for IT? As it was seen that often the opportunities would be valued over this restriction this question needs to come to play if an opportunity cannot be seized. However this restriction needs to be considered also after the obstacles and valued according to scenario in regards to the opportunities. As a

rule, if the value of this restriction is high or there are no opportunities in storing the data on external servers and the customer has large IT resources, the data should then be stored on the customer's internal servers. If, however, the customer in this case has no or small resources for IT, the external servers should be used as this brings forth more benefits of Software as a Service to the customer.

As explained by Sun et al. (2008) both customization and configuration are possible in a Software as a Service environment. However, configuration is a more cost effective way to tailor the software product to the meet the customer's needs in a multitenant environment. Also customization cannibalizes a lot of the benefits of Software as a Service as it requires changes to the source code. This is very costly and would require software maintenance separately for each customer. (Sun et al. 2008.) Customization and configuration also came up in most of the focus group interviews. It was stated that often the less customers there are, the higher the pressure to customize is. However it was stated that most of the tailoring needs could be solved with configuration, as long as the target market is fairly homogenous and the customers' needs for the same software do not vary too much. It was also suggested in by one of the participants that:

> "*Customization should be offered as an option to the customer but it should be controlled through pricing.*"

Meaning, that in offering a new product the customization should be offered as an option but the price should therefore be structured so that it would encourage the customer to go for configuration instead. It must also be remembered that the price should cover the increased costs of having to maintain the customer's software separately. Xin & Levina (2008) support the fact that customers who require customization would rather go for the on-premises model. In an interview it was also stated that in order to achieve an acceptable level of configurable choices the vendor needs to have a good understanding of the target customers and the customers' needs.

Does the interface need customization or configuration? This question is the only question in the framework that does not have a yes or no answer. It is also not quite as straight forward as stated in the framework. However, to simplify the decision model it has been stated that if the software needs customer specific customization the better option would be to store the data on the customer's servers as it would be more cost effective than to generally offer customized software on the vendor's servers. If the customer's needs can be met through configuration within the software the opportunities and benefits offered by the externally hosted software can be met.

4.1.4. The delivery models

Figure 7 depicts the last subgroup showing the final delivery models and the final questions that determine the delivery models. These questions come to play only after

the decision of whether, internal or external servers should be used for data storage and running the software. The databases act as connectors of the three other subgroups to the final delivery models.

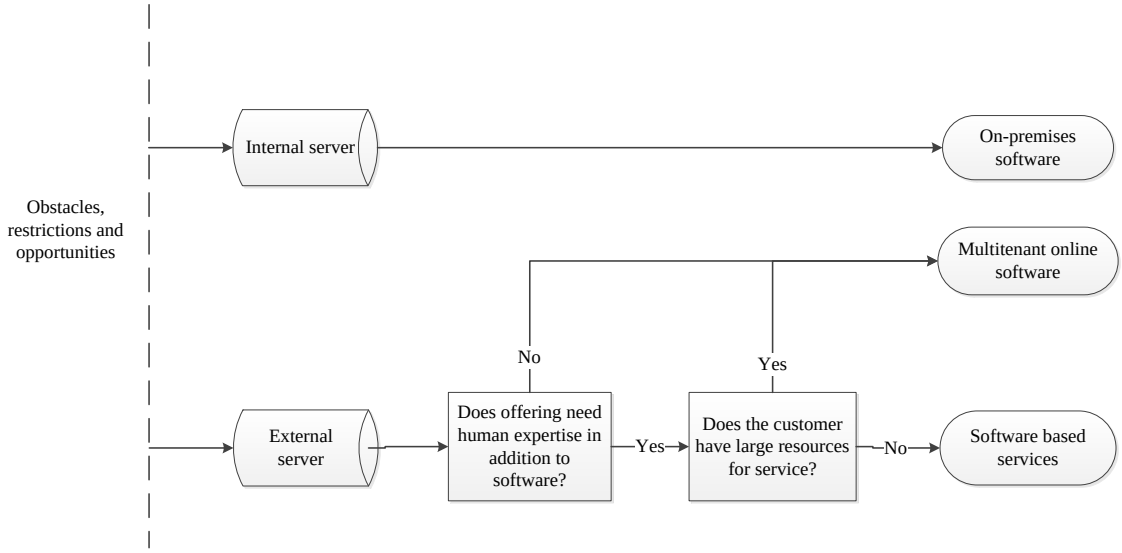


Figure 7*. Delivery models subgroup depicts the final delivery model options and the final questions after the database selection that determine the final model.*

As shown in Figure 7 running the software on the internal server only ends up in one delivery model, the on-premises software model. The external server on the other hand has two options the multitenant online software or otherwise known as the Software as a Service model and the software based services. The multitenant online software was chosen here as a term to better demonstrate that this is the delivery model that is related to the Software as a Service business model.

The basic difference between the two presented delivery models for the external server is that in the case of the multitenant online software the customer operates the software whereas in the software-based services the software is usually operated internally in order to offer services triggered by the output from the software. In order to see which delivery model the software product is better suited for, the framework includes two questions that help determine this.

Does offering need human expertise in addition to software? In two of the interviews the topic of what distinct aspects determine when Software as a Service and software-based services should be used were discussed with more detail. The biggest distinction was the need for human expertise in order to reach highest potential of added value. It was proclaimed by one of the participants in a focus group that:

> "*Especially big, expensive, remote and critical expertise should be offered through software based services.*"

In conclusion, if the human expertise is not required the decision tree leads to multitenant online software and if it is required it leads to the second question.

Does the customer have large resources for service? It was also stated in one of the interviews that in addition to the size of the resources for service it has to also be considered what kind of expertise is required and to what extent as well as what are the customer's capabilities to meet these needs of expertise on their own. Here the resources for service within the customer's organization play a role in whether the customer would rather outsource the service to the vendor or rather use the software themselves and take care of the additional services internally. Of course, this is also not quite as straight forward because even with large resources, the needed expertise can be so specific that it needs to be outsourced regardless of budgeting issues. However, in the framework, in order to simplify: if the answer to the question is yes, the multitenant online software is the better option. Whereas, if the answer is no, the software based services should be offered.

In addition, it was mentioned in the interviews that it should be looked into weather the multitenant online software could be used as a method for enabling service sales. Therefore not strictly turning into software based services but utilizing the Software as a Service to trigger additional sales in service. In most interviews specifically Software as a Service was seen as a prominent enabler for service sales but in two of the focus group interviews also on-premises software was seen as having the same opportunity.

4.2. Revenue model

The revenue model decision chart has two starting points based on whether the software is run on the client's servers or the vendor's servers. The ending points of this decision tree depict the different revenue model options some are exclusive to the starting points but there is also an option that is possible for either starting point. Yes-no –question flow determine which possible revenue model might suit best in each case. The questions have to do with the targeted customer as well as aspects of the software itself. In this chapter the revenue model decision chart is explained by dividing the chart in two based on the starting point.

4.2.1. The on-premises software

In Figure 8 the decision chart for the revenue model is depicted, showing the options and relevant questions if the software is run on the customer's servers as on-premises software. Here the first question is related to the customer, whereas, the rest are related to the characteristics of the software product.

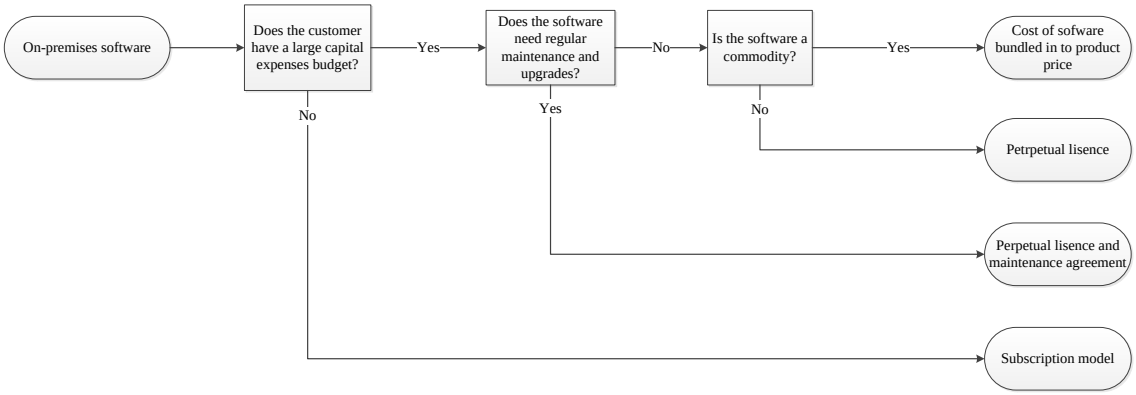


Figure 8. *The revenue model decision chart for software run on customer's servers.*

Waters (2005) states that restrictions in the customers capital expense and operational expense budgets might lead the customer into choosing a Software as a Service type of business model as traditionally the Software as a Service business model operates with a subscription type of revenue model without any initial investment. Ferrante (2006) and Konary et al. (2004) both describe the subscription model also for the on-premises software as a solution for getting rid of the large onetime expense by dispersing it over time and thus converting it into an operational expense. In literature the capital expense is seen as a hindrance that can be solved through converting the payment into a subscription model. For the vendor the subscription model does create a more steady cash flow and is therefore often preferred (Ferrante 2006; Konary et al. 2004). The capital expenses versus operational expenses conversation was also present in all the conducted interviews, often stating that most of the time customers tend to prefer operational expenses over large capital expenses. However, it was stated that not all customers prefer operational expenses over capital expenses. It was stated through an examples in one of the interviews:

> *"Some organizations work with large capital expense budgets. Such as government organizations or other companies that have big investments like power plants."*

Does the customer have a large capital expenses budget? This question distinguishes the customers that prefer or perhaps even require larger investments rather than continuous operational expenses based on their budgeting practices. If the answer to this question is a no a subscription model should be used. The subscription model is more beneficial to the vendor as it smoothens out the cash flow. The subscription model can also be viewed as a way to eliminate separate maintenance agreements (Cusumano 2008). If the customer does operate with large capital expenses budget, the next question comes to play.

Traditionally the perpetual license does not include an automatic right for upgrades and the upgrades must be bought in separate patches (Konary et al. 2004). Ferrante (2006) states, that it is very common, for the vendor to require the customer to purchase maintenance of the software product for a predefined time period. This was also backed by the discussions stating that very often industrial software is sold as a perpetual license together with a mandatory maintenance agreement and the option of purchasing upgrades or patches separately is not even an option. Konary et al. (2004) state that the maintenance fees provide indication of the future revenues by smoothening out the large differences in revenue streams that the perpetual licenses may cause, thus making it a prominent option for vendors.

Does the software need regular maintenance and upgrades? As stated earlier, vendors often require a maintenance agreement to be signed in parallel to the sale of a perpetual license. Therefore, if the software requires regular upgrades or other maintenance it is most profitable to opt for the revenue model where a perpetual license and a maintenance agreement are offered bundled together. If however for some reason the software needs no upgrades or these needs occur very infrequently the decision tree leads to the next question.

In one of interviews it was stated that often customers may not view the software price as a large investment especially compared to the investment made into hardware products possibly from the same vendor. This is a function to consider especially with industrial software that is directly connected to hardware products or systems, expressly if these hardware products or systems are delivered by the vendor's organization. The customers may often view that the software is a commodity or a necessity and viewed as an inseparable part of the product or system.

Is the software a commodity? If the software is viewed as a commodity and the customer does not see the software to bring enough added value the cost of the product should then be bundled and sold as a part of the physical product or system sold by the same organization. However if the value of the software is visible to the customer outside of the hardware it functions with a perpetual license should be offered.

4.2.2. The multitenant online software

Figure 9 depicts the decision tree for finding the best suitable revenue model when the software is run on the vendor's servers. The questions leading to the best suited revenue model are related to the software characteristics as well as the use of the software product.

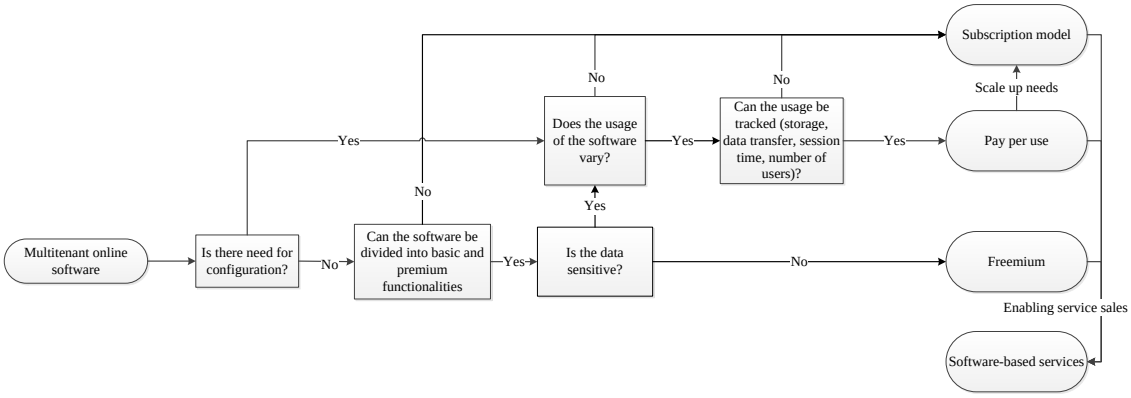


Figure 9. *Revenue model decision chart for multitenant online software.*

Sun et al. (2008) define configuration as a less costly option for tailoring a software product. This however does not mean that configuration has especially low costs. In their study they define many levels of configuration and each requires a different amount of planning and costs in different stages of the software product's lifecycle. (Sun et al. 2008.) The level of configuration is relevant to the level of costs committed to the development and maintenance of the software product.

Is there need for configuration? As stated earlier the need for configuration is relevant to the costs of the software product. This question is especially relevant to determine whether a freemium revenue model can be viewed as an option because according to Osterwalder & Pigneur (2010) with the freemium revenue model it is important that the maintenance costs of the product remain low. However, this is not the only question that needs to be considered for the freemium revenue model to be viable. If configuration is required the decision model leads to the next question that considers the other revenue model options. If there are no configuration needs the decision model leads to the next question that determines whether a freemium model is a possible option.

Osterwalder & Pigneur (2010) describe the freemium model as a marketing model. Also Shapiro & Varian (1998) describe the freemium model as a good way to attract attention to the product. According to Teece (2010) it is divided into free and premium functionalities. These free functionalities are then covered by the revenue streams acquired from the premium customers (Osterwalder & Pigneur 2010). In the interviews the freemium model was also seen as a viable option for industrial software products. Many scenarios were suggested where the freemium model may be profitable. The main idea was that the basic functions such as data from sensors could be offered for free through the software but the actual analytics or reports would only be generated as premium functionalities.

Can the software be divided into basic and premium functionalities? Where the costs and therefore configuration needs may affect the profitability of the freemium revenue

model, this question defines whether or not the freemium revenue model is at all possible. Therefore, if the functionalities can be divided into basic and premium functionalities the next question should be considered. If this distinction cannot be made the subscription model should be chosen.

Murphy (2010) states in his paper about freemium models in a Software as a Service concept that especially when the software utilizes data that is sensitive to the customer, the customers will be apprehensive about adopting the model. His statement comes from the idea that the freemium model makes customers question the safety of the software. (Murphy 2010.) Also in the paper by Anderson (2009) it is indicated that free offering makes the customer prone to doubting the quality and safety of the offering.

Is the data sensitive? As stated before the sensitivity of the data needs to be considered as this might affect the trust the customer has in regards to the software. Here it should be considered in general whether offering the software with the freemium revenue model somehow damages the value of the software or the reputation of the software or vendor. If the answer to the question in this decision point is no, the freemium revenue model can be offered. If the answer is yes, the framework leads to the next question.

Ojala (2013) states that the problem of the pay-per-use model for the customer may be unpredictable operating costs. Konary et al.(2004) also brings this problem to the vendor's side by stating that the model may also make the revenue streams lower and more unpredictable. In one of the interviews it was stated that the pay-per-use model would be especially suitable for situations, where the use of the software would vary, especially in the time intervals of how often the software is used. If for example the software would only be needed for rare projects or if the need for the use of the software is unpredictable, it was seen that the pay-per-use revenue model might be best suited for said software. This however is not the only requirement for the software for the pay pay-per-use model to be suitable. The Subscription model on the other hand is more suitable the more the software is used. The irregularity of use does not rule out the subscription model as the best viable option but, as Ojala (2013) proclaims, in the subscription model the customer pays for the use, whether or not the software is actually used. It can then also be stated that the price also does not change as the amount of usage grows.

Does the usage of the software vary? If the usage does not vary the subscription model should be considered as the best option. If on the other hand the usage is irregular the decision model leads to the next defining question.

The pay-per-use model requires the monitoring of the defined unit that the payment is based on. The unit can be based on the number of transactions, units of time or the times the application is used. (Ojala 2013). This was also a topic that came up in three of the

interviews. If the usage cannot be monitored with acceptable resources and costs, the pay-per-use model cannot be an option. An example of report generation was given to illustrate a possible and attractive use of the pay-per-use method.

Can the usage be tracked (storage, data transfer, session time, number of users)? When tracking is not possible, the subscription model is a more viable option. If the usage can be tracked, the decision model leads to the pay-per-use model to be considered as the best suitable option.

In the Figure 9 there is an arrow leading from the pay-per-use revenue model to the subscription revenue model option. The arrow states "Scale-up needs". This addition comes from one of the discussions. As stated before, the irregularity of use may make the pay-per-use model a more suitable option. If the need for usage changes and the use becomes more regular and predictable a scale-up possibility should be offered to the customer. In other words, an option for the customer to be able to switch to the subscription model. This makes the revenue flows more predictable for the vendor and the costs more predictable for the customer.

From all three revenue models there are arrows leading to a fourth ending point called "Software-based services". The arrows state "Enabling service sales". In all the interviews conducted the possibility to use the Software as a Service business model to enable service sales is prominent and should always be exploited if at all possible. With the software based services the possible revenue models are not described as they exceed the scope of the study because this could include all possible revenue models for service jobs in general. As stated in one of the discussions:

> "*Pricing should be based on value. With software based services the fact that there is software within the process of delivering the service does not make a difference to the customer value.*"

What did not end up in the framework, but was brought up a couple of times in the different interviews was the concept of performance based pricing. It was brought up as an option for the software based-services pricing model but it was soon concluded that it could not be excluded as an option for the other delivery models. However, in the end it was left out of the framework as performance based pricing is not strictly a revenue model but rather a pricing model that can be applied to all the different revenue model options. An example was given in one of the discussions as a basis for consideration. The idea of offering software for energy analysis and the pricing would be based on how big savings in energy would be reached due to using the software. In the example it was stated that if the promised performance improvements or savings were not reached, the price would be a predetermined sum that was much lower than if the promised savings were reached.

It was also stated that the problem with this pricing method would be that the causality between the software functions or outputs and the savings or improved performance need to be clear. If the causality cannot be proven the base from the entire pricing method disappears.

The performance-based pricing is an interesting concept and should be considered further. However, the subject is too vast for this thesis alone and is therefore out of scope.

4.3. Refining of the framework in the supplementary interview round

The previously explained initial framework was refined by conducting supplementary interviews, where the frame work was gone through in detail to identify mistakes or weaknesses. In this chapter the changes made to the framework are gone through in detail with explanations from the interviews. This chapter will not look into the areas where no changes were made. In the figures the areas that were affected due to the changes or specific remarks are marked in red.

4.3.1. Repositioning customer related obstacles

In Figure 10 the new version of the obstacles subgroup is presented. As mentioned the changes are marked in red. Unlike before, the changes will be gone through from the bottom of the figure moving up. This is due to the fact that the change of position of the now last obstacle question initiated the moving of the second last question.

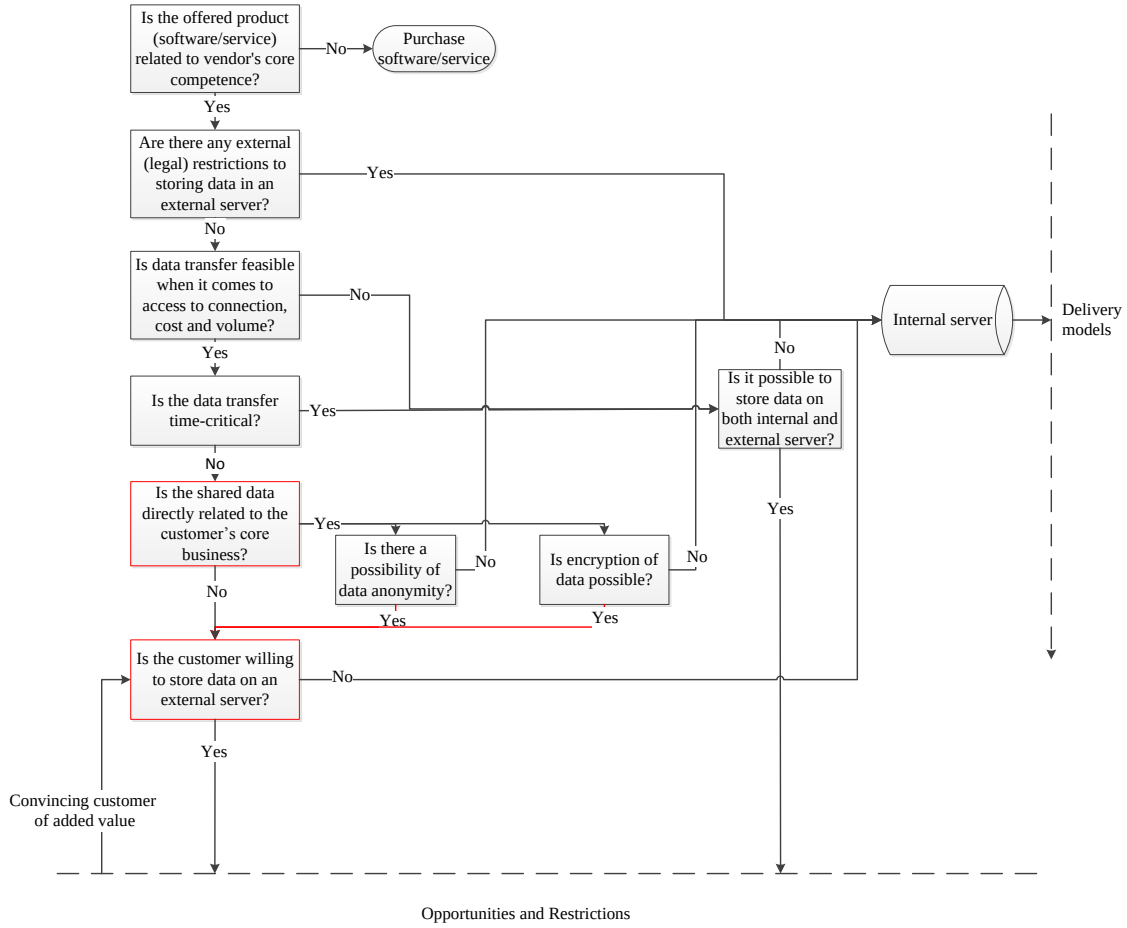


Figure 10. *The refined version of the obstacles subgroup with the parts subject to change marked in red.*

Is the customer willing to store data on an external server? This question initially held its spot as the second question in the decision chart. The initial idea came from the notion of the importance of the customer in business models and especially business models related to service. However, in all of the supplementary interviews it was stated that the willingness of the customer does not make any difference, if the external factors unrelated to the customer's inclination to share the data do not allow the data to be shared. Another issue that came up was related to the feedback loop from the opportunities suggesting convincing the customer of the opportunities that storing the data on an external server might bring. It was stated that there is no point in going through the other questions again, as the answers to those questions would not change, regardless of the opportunities. Therefore, the question of the willingness of the target customer to store data on an external server was moved to be the last question with an amenable answer leading to look into the opportunities. A declining answer despite the opportunities would lead to storing the data on internal servers.

Is the shared data directly related to the customer's core business? As the change of the previously mentioned question was agreed upon in one of the interviews, it was remarked that the question related to the customer's core business is also something that is unrelated to the external issues. The other obstacles cannot be influenced, whereas the question on customer willingness can be gone around as can the core business issue. Therefore the question of the data being directly related to the customer's core business was moved to be the second last obstacle. If the answer to this question was yes the answers would follow the same pattern as in the initial framework and a negative answer would lead to the question of customer willingness.

The change of moving the question of the data relating to the customer's core business also affected changes to the questions related to data anonymity and encryption. Previously a positive answer to these questions would lead to the question of the feasibility of data transfer. The move of the core business question down the decision chart affected where the positive answers to the questions on data anonymity and encryption would lead. In order to avoid an unnecessary and redundant loop the affirmative answers in the final framework lead to the question about the customer's willingness to store data on external servers.

4.3.2. Clarification to the tailoring related restriction

Figure 11 shows the restrictions subgroup with the question regarding tailoring: "*Does the interface need configuration or customization?"* Unlike in the previously explained case there were no visible changes made to the decision chart. The change however is to the scope of the question. The figure was added to visualize where in the framework the change has affected.

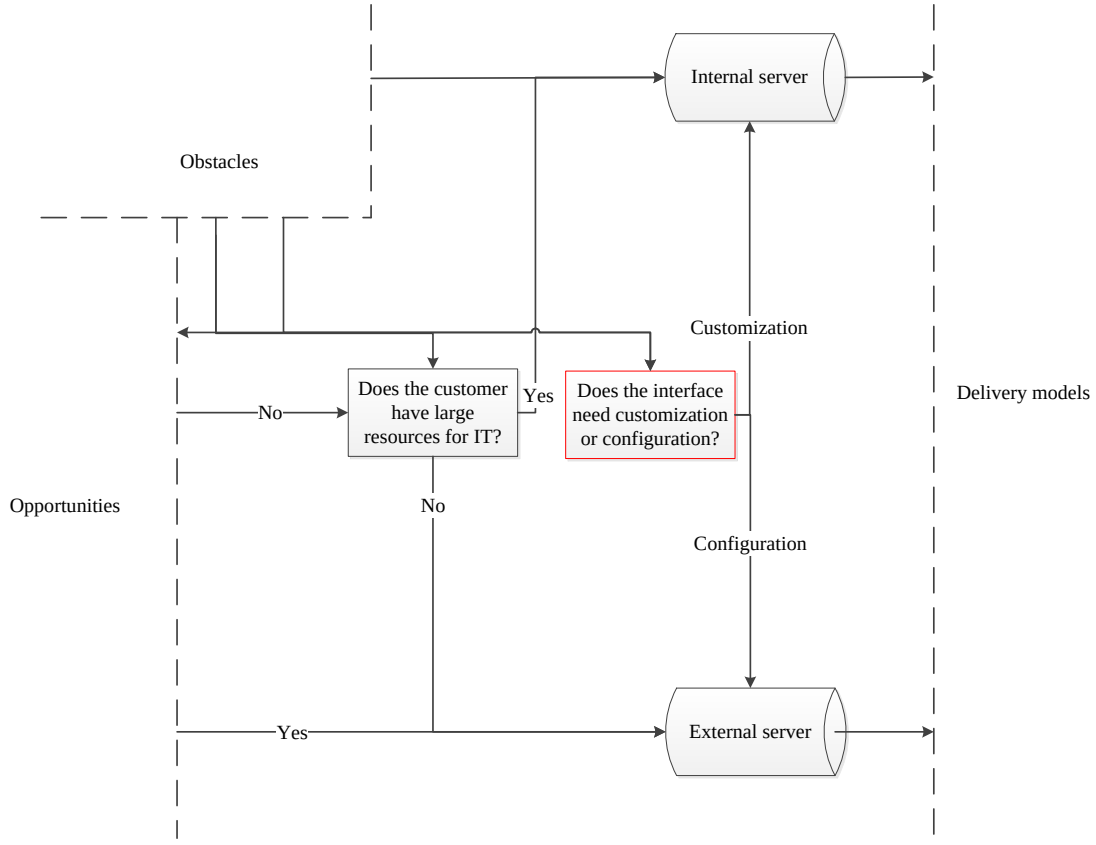


Figure 11. *The figure shows the refined restrictions subgroup with the question marked in red that was affected by changes.*

In one of the interviews a suggestion for an addition to the soft criteria of the opportunities was made. It was proposed that a question should be added stating: "*Can the application software be used across different customers or industries?*" However, in a later interview this was discussed and it was considered that rather than an opportunity, this should be fitted within the restrictions and more precisely the question of configurability and customization. It was stated that this was rather a question within a question, urging to consider if the same software could be offered to different customers and industries through customization. This would very much widen the range of the question "*Does the interface need customization or configuration?*" Not to only look into the customer needs but also the entire target market as well as possible target markets for the same software application.

4.3.3. Connecting delivery and revenue models

Figure 12 depicts the delivery models subgroup. No changes were made to the order of the questions but aspects were added to emphasize the connection between the delivery model and revenue model frameworks.

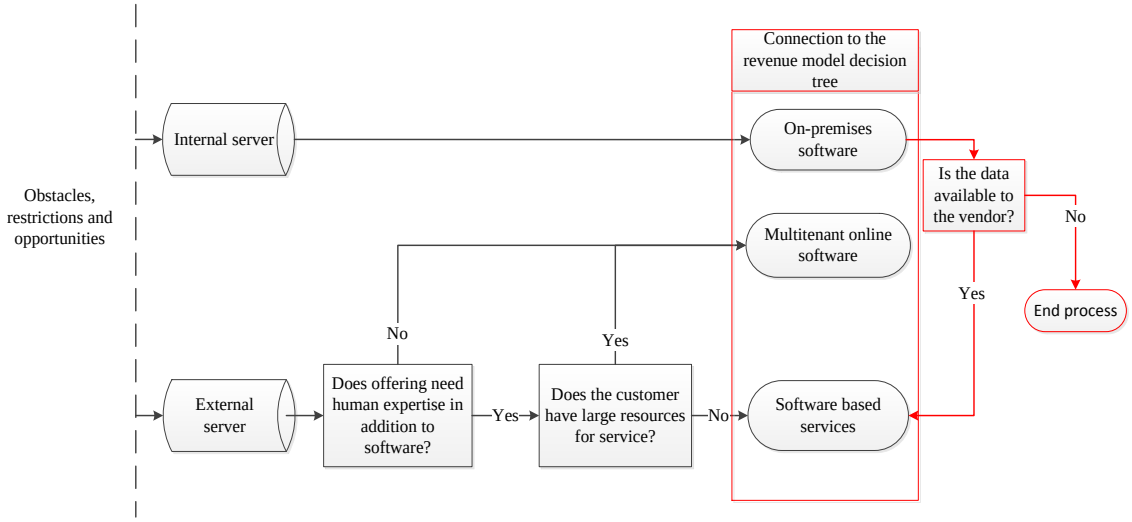


Figure 12. *The final delivery models subgroup where in red the connection to the revenue model decision chart is emphasized.*

As stated before, in the initial framework the connections of enabling service sales through the other two delivery models should be considered. It was, however, decided that this side is represented in the revenue model decision chart as it was thought that in general the delivery model of the software should not make a difference on weather software based services could bring revenues to the vendor or not. However, in a later interview it was stated that in fact the delivery model of the on-premises software should be connected to the software based services as the delivery model does make a difference. In the case of multitenant online software the software based services can be generated easily as the vendor has access to the outputs of the software. In the case of on-premises software it is essential that the vendor have access to the outputs of the software. Therefore a decision point was added: "*Is the data available to the vendor?*" If the data is available, software based services can be offered but if not, the process ends and only on-premises software is offered. (Figure 12.)

The connection between the delivery model decision tree and the revenue model decision tree needed to be emphasized. Within the box in Figure 12 the two first endings, the on-premises software and the multitenant online software or Software as a Service, represent the starting points of the revenue model. Software-based services, on the other hand, represents one of the ending point in the revenue models.

4.3.4. Enabling service sales through on-premises software

In the Figure 13 the changes to the revenue model have been marked in red. These changes are also related partly to the previously explained adjustment made to the delivery models subgroup in the delivery model framework.

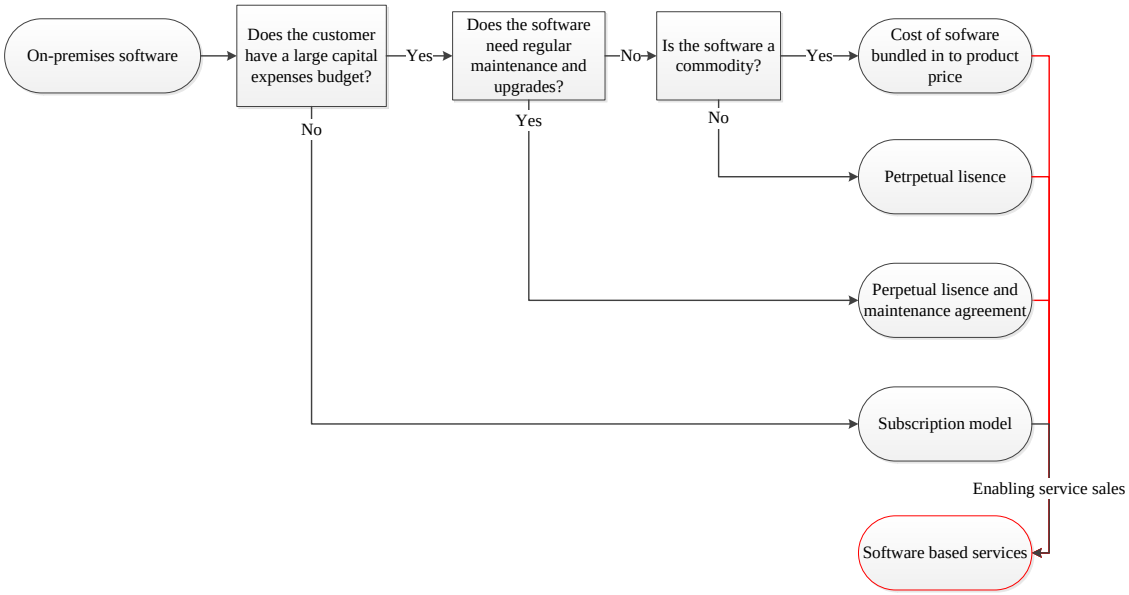


Figure 13. *The final revenue model chart for the on-premises software with the additions marked in red.*

In one of the interviews it was stated that also on-premises software can be used to enable service sales even though the software and the outputs it gives are more difficult to access by the vendor. This may however imply more proactive sales in order to be able to deliver these software based services. Due to this the connections were added from the on-premises software revenue model options to the software-based services as shown in Figure 13.

4.3.5. Redefining connections in multitenant online software revenue models

In the Figure 14 the multitenant online software or Software as a Service side of the revenue model is depicted in its final form. One major changes is introduced based on discussions had in the supplementary interview round and another view at literature. The change is marked in red.

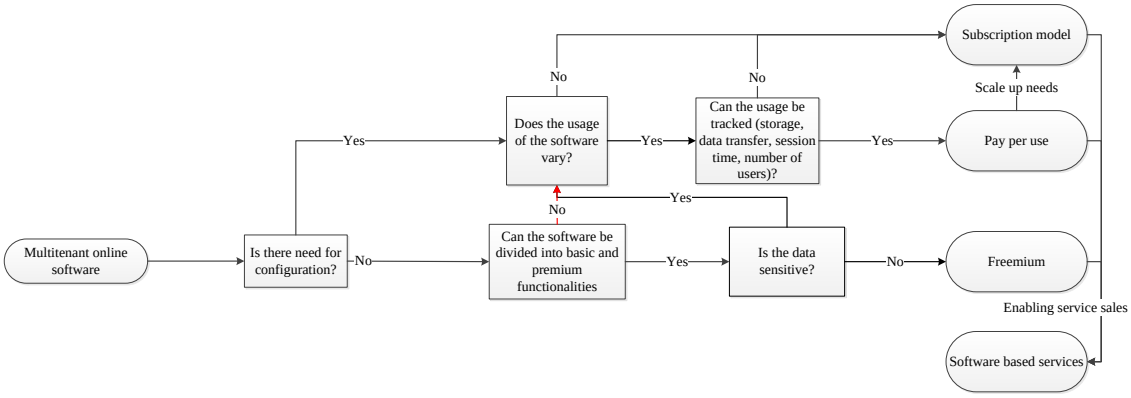


Figure 14. *The final version of the multitenant online software side of the revenue model decision chart with the changes marked in red.*

As seen in Figure 14 there is only one change in the Software as a Service part of the revenue model decision chart. However this change is a significant one. In one of the interviews it was pointed out that the fact that there is no configuration need for the software should not exclude the option of the pay-per-use model. It turned out that in the case company already a form of pay-per-use model was used for a standard software with no configuration options. Therefore, it was decided that from the question "*Can the software be divided into basic and premium functionalities?"* rather than leading directly to the subscription model if the answer to the question is no, the model leads to the question "*Does the usage of the software vary?*"

4.4. Validation of the framework

The first validation workshop did not cause any changes to be made to the framework. In the first workshop mainly the future use of the frameworks was discussed and it was stated that the framework will be utilized in a project looking into possible business models for industrial software service solutions. This project will then look into the topic further and in more detail. Also the future project will take a more ABB specific view on the business models and the questions that have to be regarded when looking into the different business models.

The second workshop also initialized no changes to the framework. In this session three main questions were asked. 1) How can we use the framework to test the current ABB portfolio? 2) Are we oversimplifying the categorization customer-specific versus standard offering? 3) Where does the data reside, is there a breakdown of data sources in ABB? Questions 1) and 3) are going to be answered further in the future project conducted in ABB as the time and resource restrictions for this thesis do not allow this detailed view on the matter. Question 2) is discussed shortly in the discussion part of the thesis.

All in all, the weak market test can be deemed successful. As the definition of the weak market test states the test is passed if the examined construction is either in use in the market or someone wants to use it (Hakkarainen 2006, p. 160). The first workshop proved that the framework will be utilized in a future project to identify business models in regards to industrial software service solutions. The second workshop more over validated the content of the framework to be correct when it comes to interpreting the conducted interviews.

The final delivery model framework can be seen in appendix 3. The framework brings together the obstacles, opportunities restrictions and the final delivery models in one cohesive decision tree. The decision tree represents the path to finding the best suited delivery model when the target customer and the value proposition, meaning the requirements of the software, are known.

The final revenue model decision tree is represented in appendix 4. In the final decision tree the previously separate decision trees for the on-premises software and the multitenant online software are brought together. This exemplifies better the connection between the delivery model framework and the revenue model framework. The delivery model decision tree defines the starting point of the revenue model decision tree allowing the revenue model decision tree to have two starting points. Another reason to connect the two separately presented revenue model decision trees was the common revenue model, the subscription model, which seamlessly combines the two decision trees into one.

5. DISCUSSION

The discussion chapter first examines the possibilities of use that the previously presented framework offers. After this each business model is looked at separately pinpointing the things that need to be considered when implementing each business model. These chapters also determine whether the business models are worth implementing in the case of an engineering company and especially ABB. After this the opportunities and requirements of the business models are looked at cohesively following an action proposition for ABB and a description of the next steps.

5.1. Use of the business model selection framework in practice

The framework was built as a decision tree that can be gone through from start to finish, ending up with the best suited delivery model and finally the best suited revenue model. The delivery model framework is the first part of the entire framework and according to which delivery model is best suited the starting point of the revenue model framework is determined.

The biggest and most obvious opportunity of the framework is to use it as a decision model for potential software products. The framework helps to figure out the best suited delivery model and revenue model depending on the target customer and type of data that the software needs to serve its purpose. In this sense the framework is used to build a complete business model.

As stated before the Tsvetkova & Gustafsson (2012) have defined four components of a business model. The first is the value proposition, meaning the benefits that the customer receives from the offered product or service. The second component consists of the capabilities, meaning how the value is delivered to the customer. (Tsvetkova & Gustafsson 2012.) The capabilities in this thesis are summarized as the delivery models because the different capabilities behind the software define the actual delivery model of the software or service. The third component according to Tsvetkova & Gustafsson (2012) is the customer and the fourth the revenue model.

As the framework offers a decision chart for finding the best suited delivery model and revenue model the prerequisite of using the framework in the initially intended way is that the software product itself is well determined. To be more specific the value proposition of the software must be known. This includes the functions and the outputs

of the software but also the requirements that the software sets. The requirements would include the type of data and the intervals that the data is needed or the amount of configuration or customization the software may need.

The framework can be viewed as being very data-driven. This can be explained by the fact that the company, where the empirical part of the study was conducted is an engineering company rather than a software vendor. As the vendor in this case is an engineering company, the best value in the offering can be derived from the knowledge of the products, systems and processes offered by the company as well as the industry knowledge. Therefore the data from the products, systems and processes is what can be stated to be in the core of the value proposition. The data is then transformed by the software in to the actual value, weather that then be the outputs of the software or the services built on top.

The data-driven approach is touched upon in various literature but the concept is not emphasized in the studies of the actual business models. Schnjakin et al. (2010) merely mention that legal restrictions may occur if data needs to be collected from the customer. In other cases the main focus of the studies is delivering the data to the customer rather than collecting data. Sun et al. (2008) and Dubey & Wagle (2007) emphasize the online connection between the vendor and the customer but their viewpoint is again the information flow from the vendor towards the customer. Weinhardt et al. (2009) do consider the data transfers when talking about cloud computing or Software as a Service, stating that service level agreements could encourage firms to utilize the cloud even for mission-critical industrial services. Weinhardt et al. (2009) does not specify the data stored on the cloud or how it is collected. The perspective of data storage comes up in the literature of Software as a Service in cases such as Armbrust et al. (2010), Sun et al. (2008), Waters (2005) and Weil (2007). These, however, present a very narrow approach to data collection and do not consider the value added and services that can be attained through analyzing and processing the data.

The data-driven approach is one of the main indications of the industrial and service aspect presented in the thesis. The collection of data is an important point in this. It is something that restricts the use of external servers and if the collection of data is important the customer’s servers must be utilized. The most important point, however, would be that the data-driven approach brings new opportunities when using the external servers to deliver the software or services. These opportunities can be seized through data aggregation and combinations, as well as utilizing the same data to create different value propositions to perhaps different customers, as presented in the results.

The other prerequisite is the knowledge over the target customer. The targeted customer must be well know from the knowledge over their processes to their organization structure and budgeting practices. In addition to this the environment in which the customer as well as the vendor operates needs to be well know.

The customer segment or the specifically targeted customer must be well known in many aspects. Tsvetkova & Gustafsson (2012), Morris et al. (2005), Chesbrough & Rosenbloom (2002) and Shafer et al. (2005) all emphasize the targeted customer or customer segment as being a vitally important part of the business model. The size of the customer company or the amount and type of users must be known to find the best suited business model. In regards to the revenue model Ojala (2013) finds larger firms as more willing to go for the perpetual license model and therefore the on-premises software. Xin & Levina (2008), on the other hand, state that the more users the customer has the more likely they are to adopt Software as a Service instead of the on-premises software. Waters (2005) emphasizes the heterogeneity of the application users. All of this information requires deep and comprehensive knowledge over the customer's or at the very least customer segment's traits.

Sun et al. (2008) comprise an ample list of traits of the customer that effect the needs the customer has in regards to the software. These traits, the customer's industry focus, behavior, culture, strategy, regulations as well as the product offering, are all things that need to be known of the targeted customer or segment. The traits are listed in consideration about the tailoring needs that the customer may have in regards to the software. (Sun et al. 2008.) However, these traits are aspects that need to be considered in other situations as well. The product offering and the strategy as well as the industry focus may affect the core business of the customer or the willingness to share data. Therefore affecting which server can or should be used. Behavior and culture affect the willingness to share data and regulations affect the external restrictions and this leads to restrictions on the server options. These are just some examples, but the conclusion that can be made and that is backed by various literature, cited previously, both from the business model side as well as on the on-premises and Software as a Service side.

When the target customer and the value proposition of the software are known the framework helps complete the business model with the best suitable delivery and revenue models. This, however, is not the only way that the framework can be used. The customer referred to in the framework can also be directed to a specific customer instead of an abstract target customer. In these situations the framework can be used as a tool to support sales.

When using the framework as a sales tool, the different end business models need to be known and whilst approaching a potential customer and acquiring information of this customer, the sales person can identify different aspects of the customer that relate to

the framework. This way the sales person can identify the best suited product to offer the customer.

The main contributions in the industrial sense for industrial software service solutions is the data-driven nature explained earlier. The value proposition in the industrial software service solutions is most often very closely related to the customer's processes or systems. This is an aspect that does not appear in the literature when viewing these business models in the commercial viewpoint. The customer also differentiates the industrial software business models from regular software business models. The industry focus, product offering and strategy play a much larger role in the willingness and ability to adopt the externally hosted business models, especially when the vendor is an engineering company and therefore the core competence of the vendor relates to the processes and manufacturing products that the customer has and the vendor also produces.

Safety and data security is a big issue especially with Software as a Service due to the external server being used. Waters (2005) states that even though Software as a Service vendors can often achieve better data security customers feel apprehensive in trusting the safety of external servers. In the framework that is presented in this thesis safety and data security is not addressed head on. Some questions, such as the questions about data anonymity and encryption, touch on this subject but in a very one sided manner. The safety and integrity in regards to data in this framework are seen as a prerequisite that needs to be in order in order to be able to offer the software and services. This could cause challenges if the safety issues are not confronted before using this framework.

Benlian & Hess (2011) state that security is the main perceived risk by IT executives in regards to adopting Software as a Service. They imply though that this risk is related more to the contracts made with the vendor rather than actual security breaches in the software. (Benlian & Hess 2011.) Waters (2005) sees the problem as a perceived issue in the software safety being eminent as well, even though, data security can often be achieved better and more cost effectively by the vendor. Ojala (2013) adds another safety benefit to Software as a Service by stating that the possibility of piracy goes down when the software is run on the vendor's servers.

Even though safety is viewed as a prerequisite for the framework it is not enough that the sufficient safety measures are taken. A trust among the customer and the vendor need to be achieved. As stated, the perceived safety is the main issue in adopting the Software as a Service business model. If there is a lack of trust, in regards to safety, with the customer the options limit to offering the on-premises software but even then the situation is not optimal. Therefore, to choose any of the business models the safety and security measures must be in place and sufficient but also the relationship with the customer must be well kept to ensure that also the perceived security is sufficient.

In addition to the previously mentioned opportunity to find the best suited delivery model and revenue model for a specific target customer and a specific software product, the model can also be looked at in another way. Instead of starting from the beginning of the model the framework can be looked at from the endpoint just as well. Looking at what should be considered when a certain business model of the three alternatives is wanted. In this case the framework has to be gone through backwards to find out what needs to be taken into consideration and what functionalities need to be fulfilled.

Most of the literature on the software business models consider either the delivery model or the revenue models but do not consider both comprehensively. Waters (2005), Foley (2004), Buxmann et al. (2008) and Benlian & Hess (2011) for example have very narrow views on the revenue models presented for the software business models. Waters (2005) for example sees the Software as a Service revenue model as the rental or utility model and the on-premises software with the license as the predominant revenue model. Ojala (2013) on the other hand only looks at the differences of the revenue models but does not go into detail on the differences of the delivery model. This thesis brings these studies together by examining both distinguishing factors in the delivery and revenue models in the industrial scope.

The framework can also be looked at through what are the requirements and opportunities of each business model. This analysis will demonstrate what must be taken into consideration when aiming for each business model. In this type of analysis check lists can be formed from the questions of the framework identifying requirements or opportunities of each business model or delivery and revenue model separately. The business models will be looked at in the following chapters.

In conclusion, the framework shows an important aspect of the industrial software service solutions. The data-driven nature of the software and services is the key, especially in the case of an engineering company being a vendor. The framework can be used to find the best suited delivery model for a software and the best suited revenue model based on the delivery model chosen. The other use of the framework is to look at the delivery models or revenue models and look back at the things that are required for each model. The prerequisites for using the framework is that the value proposition or in other words the characteristics and requirements of the software are well determined. In addition to this the targeted customer or customer segment needs to be well known. The customer relations need to be good, to ensure that the perceived safety of the software does not suffer and the security measures need to be sufficient in each case.

5.2. Limitations of the framework

While the framework proves to be quite adaptable to different purposes, it has its limitations. The framework is based on interviews within just one company. The

organization where the study was conducted also does not yet have all the different delivery or revenue models in use in the purest sense. Therefore, the views from the interviews are somewhat one sided even though the effects of this was attempted to be minimized by choosing interviewees from different parts of the organization as well as people with a good understanding of the entire organization. With the company being a multi-industry organization the view is not limited to only one industry but nonetheless, the view within just one company may produce some biased opinions that affected what appears on the framework.

The main challenge of the framework is that it is quite complicated and may be difficult to understand without deeper knowledge into the questions. The framework poses multiple options that sometimes have to be considered in multiple phases of the framework, as well as options that must be considered simultaneously. If the answers to these differ, they may lead to separate endings. It must be understood that despite the different endings the options are not exclusive of each other.

The framework also simplifies issues quite roughly. The explanations of each question presented in the results portion of this thesis aim to soften this simplification introducing more complex considerations and background to the question. However, in some cases further analysis is needed. The restrictions subgroup is the most prominent example of this. The question "*Does the customer have large resources for IT?*" cannot in reality be taken as a simple yes or no question. A thorough analysis must be made to weigh the opportunities that can be seized through offering to run the software on the external server. Even if no specific or no significant opportunities are reached the customer may have reason to still prefer the external servers despite having large IT resources. The question "*Does the interface need customization or configuration?*" is also one that is simplified to the point that it requires further investigation case by case. Either answer, customization or configuration, do not actually exclude any option. Customization is often too costly for using the external server but such is also the case if too extensive configuration is needed. A comprehensive cost analysis should be done in each case and the costs must be weighed against the perceived opportunities.

Also, the costs are not considered fully in the framework. The framework assumes that the overall costs are tolerable in all the situations. It is vital that a separate cost analysis is made of each intended business model.

Similarly to the delivery model the most apparent challenge in the revenue model framework is that it is very coarsely simplified. It assumes that the answers to the distinguishing questions are straight forward with only two options. Every answer needs to be considered in the sense of which of the options, yes or no, is more suitable or less wrong.

The bigger challenge or limitation however is making the false presumption that whichever end result this framework gives, this result would then be the most profitable one. The framework only shows which revenue model would be best suited, when it comes to characteristics of the target customer and the software. For instance the pay-per-use may or may not be the most profitable revenue model, if this is the end result that was reached. Just as well the subscription model is a possible option and it could be more profitable.

In order to get the economically best option, an analysis of the costs and predicted profitability must be made. These analyses should be used side by side with the framework. The economical side has been considered minimally in the framework but in order to have a reliable result further analysis is imperative.

In addition to this, the literature on the different software models were all quite limited in regards to the view that was pursued in this thesis. All of the literature conveyed the perspective of a pure software company. The presented framework however attempted to bring the standpoint of an industrial company that offers software and software based services in an industrial setting and so that these services and software fit in to and support their product and service portfolio. There is some visible mismatch with this perspective and the presented literature.

This disparity was attempted to be patched with the industry view from the company, which also had its weaknesses as explained earlier. However, the current literature on the topics and the empirical study, that has the restrictions of the availability of external sources and time, attempt to balance each other out. This being said, the limitations and challenges must be considered when utilizing the framework.

With more literature on the subject published in the future the framework can be bettered gradually to show a truer vision of the framework in an industrial setting. Perhaps new distinguishing questions may come to light.

5.3. Implementing the on-premises software business model

The on-premises software is often demonstrated as the old fashioned and even the poorer option when compared to Software as a Service. However, the framework shows that in some cases it may be useful to offer the traditional on-premises. If the on-premises software were to be left out as an option completely also certain opportunities must be stripped from the portfolio.

The frame work can also be analyzed backwards, in a sense, by picking a business model and looking back the framework identifying requirements and opportunities. This

way, for example, the scenarios in which the on-premises software is opportune as a delivery model can be found.

The requirements for offering the on-premises software are quite straight forward. This can be concluded when analyzing the framework. The requirements can be derived from the delivery model decision tree. Basically, all of the decision points that affect the on-premises software relate to questions that restrict using external servers. Therefore the restrictions can be stated quite simply:

The requirements for on-premises software:

- The customer must have sufficient IT resources and infrastructure to host the software application

The only requirement for on-premises software is that the customer has the required infrastructure to run the software on site. Also sufficient IT resources are a prerequisite even if the upgrades and maintenance was bought through a maintenance agreement, because the infrastructure and platforms must also be maintained.

A check list can also be derived to emphasize the aspects that support the on-premises software model. The check list lists the opportunities and benefits that the on-premises software brings forth. The checklist for the on-premises software goes as follows:

The opportunities of the on-premises software model:

- There are usually no external restrictions to storing the data on the customer's servers.
- The software can be run offline
- Any kind of data can be stored on the customer's servers regardless of the time-critical or sensitive nature of the data
- Customers trust the internal servers easier
- If the customer has large IT resources they may be more inclined to adopting software running on their own servers
- If the software needs customization on-premises software is the less costly option

Mainly the opportunities can be derived from the obstacles and restrictions subgroups presented in the results chapter. Schnjakin et al. (2010) and Smedinghoff (2008, p. 2) look at regulations concerning the data leaving the customer site, these regulations do not apply in on-premises software. The customer should always have the right to their own data. This is why it could be said that there is practically no case where the customer could not operate software that runs internally on their servers.

Unlike Software as a Service, on-premises software needs no online connection as it is run and maintained by the customer and in the premises of the customer (Xin & Levina 2008; Waters 2005.) From this it can be deduced that one of the biggest benefits of the on-premises software can be run offline. Therefore, when the online connection is not feasible the on-premises software is by far the best option.

Also other restrictions, such as sensitivity and time-criticalness, can be overcome as they state no obstacle to the on-premises software. The time-criticalness of the data can be explained in the same way as the statement of running the software offline. In the case of on-premises software disruptions and delays in connection are less likely as they do not depend on the availability or speed of the internet connection. As Ojala (2013) states, when the risk resulting from losing the internet connection is high the internal servers should be preferred. The data sensitivity issue stems from the perceived security issues. Waters (2005) points out that Software as a Service may be able to reach better security more cost effectively but in reality the problem is often in the perceived security from the customer's perspective. The customers often relate the on-premises software as a more secure and security-wise understandable solution. (Waters 2005.)

A lot of the literature relate the customer's large IT resources to the customer's tendency to select the on-premises software over the Software as a Service option (Waters 2005; Ojala 2013; Xin & Levina 2008). This is not strictly a benefit but rather an opportunity to also reach those customers that are not willing to hand over the maintenance responsibility to the vendor. The IT resources, though, in this situation are a rather narrow view. Also other issues may affect the tendency of the customer to prefer the on-premises software. In the focus groups it was emphasized that in the industry there tend to be very traditional and old fashioned companies and especially the larger companies tend to prefer to stick with the more familiar business model.

The last item on the list states that when customization needs to be offered the on-premises software is the opportune solution because of less costs. The engineering effort is the same no matter what the delivery model is, the savings in cost base on the maintenance responsibility. In the case of on-premises software the maintenance is mainly the responsibility of the customer (Xin & Levina 2008). Whereas in the case of Software as a Service customizing would require software maintenance separately customer by customer (Sun et al. 2008). The cost savings would mainly come from savings from maintaining and running separate customer specific software. This is not quite so straight forward though, it must always be separately determined if the customization in regards to the engineering effort is worth it cost wise.

The question stated in one of the workshops touched on this matter of customization: *Are we oversimplifying the categorization customer-specific versus standard offering?* As stated earlier the on-premises software is not necessarily always customer specific.

The issue here lies in the previously mentioned engineering effort. Very often it is much more cost effective to offer the on-premises software, tailored through configuration rather than customization. The main reason why the simplified division is made in the framework is that, if the framework is used as aid in sales, the question requires the salesperson to consider the engineering effort and therefore rather a more strict division is made.

When the cost of software is bundled into the price of the hardware product, such as a production machine, the software does not strictly have its own revenue model but the revenue model depends on the revenue model of the sold product. The requirements of this revenue model would go as follows:

Cost of software bundled into hardware product price:

- The customer has a large capital expense budget
- The software does not need regular maintenance and upgrades
- The software is a commodity

It could be said that when the costs and price of the software need to be bundled in to the hardware product price, the software on its own cannot be a very profitable source of revenue. Commodity products rarely attain very high margins. However, in these cases the software is seen as a seamless part of the hardware product, such as a production robot or machine, and therefore the software should be offered based on the profitability of the attached product. This revenue model does not pose any specific opportunities but rather is something that must be done under the circumstances dictated by the requirements.

The perpetual license model is the most traditional revenue model for on-premises software. The requirements for the perpetual license model can be identified flowingly:

Perpetual license

- The customer has a large capital expense budget
- The software does not need regular maintenance and upgrades
- The software brings additional value to the customer

The perpetual license should only be offered, if the customer insists on dealing with capital expenses rather than operational expenses. The perpetual model, while possibly profitable, ends up in fluctuating and unpredictable revenue streams (Konary et al. 2004). Adding the maintenance agreement would be the preferred option as it smoothens the revenue streams and profitably locks in the customer (Konary et al. 2004; Butler 1999). Therefore merely offering the perpetual license and the separate patches and upgrades should only be offered if the maintenance agreement cannot be sold separately, if the software does not need extensive maintenance.

The perpetual license sold with a maintenance agreement is a more profitable option compared to a mere perpetual license. The check list for this option is very similar to the previous with just a minor difference relating to the need for maintenance. The check list goes as follows:

Perpetual license and maintenance agreement

- The customer has a large capital expense budget
- The software needs regular maintenance and upgrades

The subscription model is the last revenue model option for on-premises software. This could be deemed as the most desirable. The only prerequisite for the subscription model would be that the customer prefers operational expenses over capital expenses. This is also the reason why there is no point in deriving a separate checklist for the subscription model.

The framework suggests that the subscription model should be the most appropriate revenue model for the on-premises software. Cusumano (2008) views the subscription model to be a way to go around having to sell separate maintenance agreements. The contracts become more flexible and the revenue streams more predictable and reoccurring (Konary et al. 2004).

Overall the on-premises software does bring opportunities and should also be included in the software offering. Through the on-premises software customer segments that are slower movers and therefore apprehensive about newer models can be reached. The on-premises software is also the one that should be offered when the surroundings prevent other models. In these cases the size of the target market should determine if the software should be offered.

As for the revenue models the subscription model should be deemed the most desirable as it offers reoccurring predictable revenue flows and more flexible contracts as well as allows the customer to avoid large capital expenses. If the subscription model cannot be offered the subscription model should always be paired with a maintenance agreement in order to smoothen the revenue streams. Only after this, the single perpetual license and finally the software and product price bundle come in the priority list. However, it must be stated that in each case a profitability analysis is vital and should always bypass the revenue model priority list presented in this thesis.

5.4. Implementing the software-based services business model

The software-based services, while not covered by previous literature, are an intriguing subject and shows a lot of promise. The idea in this business model is to utilize the

advances in software to either create service offerings more efficiently or to create completely new services.

The requirements of the software-based services are somewhat straight forward. The framework connects the on-premises software business model and the Software as a Service business model to the software-based services business model. The only prerequisite in the framework being that the outputs from the software are available to be used. The other requirement, even though the framework does not directly visualize it, is that it must be possible to convert the outputs into services. Therefore the software-based services requirements are listed as follows:

The requirements for software-based services

- The outputs of the software must be available
- The outputs can be converted into services

The opportunities that are related to using the external servers are essential in the creation of new services. Of course, some of these opportunities are also directly associated with the opportunities of Software as a Service. However, the software-based services related opportunities exclude the opportunities linked to the benefits that the customer gets from using the software because in the software-based services the customer may not ever see any interface to the software. These opportunities present possibilities to discovering new service offerings that could not be seized otherwise or at least not as efficiently.

Opportunities of software-based services when using the external server:

- Internal co-operation has a more easily accessible platform
- The data can be used and converted to serve the needs of other target customers
- Data aggregation can create additional value and new opportunities

When the opportunities are stripped down from the ones more closely related to just the Software as a Service business model, the opportunities of software-based services are very data centric. These listed opportunities can serve as triggers in coming up with new kinds of service offerings that bring value to the customer. In short, new services can be created through internal co-operation, data aggregation and utilizing data from different sources.

Even when these opportunities are not seized the software-based services should be held as an option. However, in these situations running the software on the vendor’s servers does not make any difference to the end result. In this scenario, the software can even be

operated by the customer. As long as the vendor has access to the outputs of the software one way or another services can be offered based on these outputs to the customer.

In the previously explained situation, utilizing the on-premises software and Software as a Service offering as enablers for services comes to play. Where software-based services often require proactive sales to get the best profitability and higher revenue streams the other software offerings can be used to ease the service sales. For example certain outputs from the software may automatically generate service suggestions.

With Software as a Service this is relatively easy to enable service sales bot through the software and through proactive sales since the vendor has easy access to the outputs of the software. There is also a possibility to communicate the need or offer of services online. The on-premises software is a more difficult issue as the access to the outputs of software is not a given. Only if there is a way to retrieve these outputs, a service offering based on those can be put together.

Nonetheless, whether it be pure software-based services, or the services are enabled through the other software business models the software-based services business model is something that should be pursued. The model has so many options of new revenue streams that it should definitely be analyzed more thoroughly. Moreover, the main point to be interpreted from this analysis is that no matter what business model is chosen for an industrial software offering, the possibility to offer services on top of the software should always be considered.

5.5. Implementing the Software as a Service business model

Software as a Service is the new prominent software business model that also intrigues the industrial sector. While the basic idea behind this business model only differs from the on-premises model by the location of where the software is run and how it is distributed online as well as new types of revenue models, Software as a Service also offers new opportunities for the types of software offered.

When the only differentiator in the offering is the way the software is delivered or used by the customer, the decision on whether to offer Software as a Service or on-premises software should be done case by case according to the customer's needs. If the value in either offering is exactly the same, the customer's willingness, the perceived safety and the customer's large IT resources play the largest role in which delivery model to offer. This is also assuming that none of the obstacles described in the framework come true. The obstacles from the framework can be derived into a check list of requirements that

must come true for Software as a Service to be possible. The requirements would list as follows:

The requirements for Software as a Service:

- There cannot be any legal restrictions to storing data on external servers
- The data cannot be time critical
    - If the data is time critical there should be a possibility to store the data both internally and externally
- The data should not be related to the customers core business
    - Unless the data can be encrypted or anonymity ensured
- The customer must give consent to storing data on the external servers

The requirements check list defines what must, cannot or should not be the case in different situations for Software as a Service to be a possible offering. The indented items are clauses that if true, can bypass the requirement it belongs to. The reasons for each of the requirements have already been explained in the results chapter so there is no need to go into detail of the content.

It should, however, be stated that all of the restrictions listed are data related. Therefore, if the software that is offered as a service online does not require any automatic data input from the customer these restrictions can be ignored. In the case of an engineering company being the software vendor, however, the software is more often than not data driven. This can also be deducted from some of the industrial software categories introduced in the industrial software chapter of the literature review. From the categories introduced as an example control, coordination and supervisory systems often require data from the affiliated processes and machines to function. The data-driven nature can also be proven by the data centric focus of the focus group discussion results. Most of the issues and differentiators that were brought up, were data related. The conclusion can be made that in most cases industrial software is somewhat data driven, especially when an engineering company offers the software.

In addition to having to fulfill the previously mentioned requirements Software as a Service can offer benefits that are either easier attainable or only attainable through Software as a Service. The opportunities can be recorded as follows:

Opportunities of offering Software as a Service:

- Close contact to the customer is easy to achieve
- Internal co-operation has a more easily accessible platform
- Adopting the software is more beneficial for the customer if users are geographically distributed
- Demand uncertainty can be contained better
- The data can be used and converted to serve the needs of other target customers
- Data aggregation can create additional value and new opportunities

The opportunities check list can be seen as opportunities for both the customer as well as the vendor. The opportunities specific to running the software on an external server are both opportunities that benefit the vendor especially. The easily reachable opportunities, on the other hand, include both customer-centric and vendor-centric opportunities. Close contact to the customer is an opportunity for both the customer and the vendor. The internal co-operation is an opportunity mainly beneficial to the customer and the two remaining opportunities are mainly customer-centric. However, it could be stated that the more vendor-centric opportunities are in fact in the end also beneficial to the customer as they will receive a more diverse offering and most likely additional value. Also, the more customer-centric opportunities can be seen as opportunities for the vendor as they incite customers to adopt the offered software or services. The opportunities have already been explained in more detail in the results chapter.

It could also be stated as an opportunity that customers with small IT resources often prefer Software as a Service (Xin & Levina 2008). This statement does not quite merit a spot in the opportunities list as it does not directly bring a huge opportunity to the vendor. What could be seen as a vendor benefit though is the possibility to therefore expand the Software as a Service offering to smaller customers. This however is completely dependent on the software product itself.

The tailoring issue is also valid when implementing Software as a Service. This topic has also been touched upon earlier in the results as well as the chapter on implementing on-premises software. As stated earlier, the statement that the only tailoring that should be allowed for Software as a Service would be configuration proves to be quite harsh. The actual decision should be made based on a thorough cost analysis but as a rule of thumb it should always be kept in mind that usually it is most cost effective to rather solve tailoring issues through configuration when offering Software as a Service. This being said also the costs of the engineering effort that goes into the configuration options must be taken in to consideration.

Software as a Service has five options for the revenue model. Unlike in the on-premises model the revenue models cannot be prioritized. The framework only explains in which cases the revenue models work and reversely which scenarios prevent the use of a certain revenue model. The actual prioritization must be done by an exhaustive analysis on the profitability of each revenue model in general and case by case.

The subscription model is the most common of the three revenue models. Basically in the case of SaaS it is a possible business model in any given situation. Therefore a check list on the requirements of the subscription model for SaaS is not necessary. For the other two requirement check lists can be derived.

The check list for the pay-per-use model is quite straight forward with clauses that must come true for the revenue model to be valid in the situation. The check list for the pay-per-use revenue model states:

Pay-per-use

- The usage of the software varies
- The usage can be tracked

Finally the freemium revenue model consists of a clause that cannot be true and a clause that must be valid for the freemium model to be usable. The check list for the freemium model lists as follows:

Freemium

- The software does not require configuration per customer
- The software functionalities can be divided into basic and premium functionalities

The check lists are easily derived from the framework but serve a different purpose. The framework is used to see what revenue model is best suited in a specific situation. The check lists on the other hand define what aspects need to be considered or must or must not be present in the situation for a certain revenue model to work. In this way the initially quite straight forward framework proves rather versatile.

5.6. Opportunities and requirements of the industrial software business models for the engineering firm

The previous chapters 5.2, 5.3 and 5.4 explained the implementation of each software business model in detail. The chapters explained what needed to be considered when implementing said business models.

The requirements and opportunities of each business model were mainly derived from the delivery model framework but also from analyzing the literature presented in chapter 2. The requirements list the things that must or must not be in place for it to be possible to implement the business model. The opportunities, on the other hand, are the business model specific benefits that each business model brings either to the vendor or the customer. A summarized table of the requirements and opportunities of each business model can be found in appendix 5.

The requirements and opportunities listed in the table in appendix 5 are specific to requirements and opportunities of precisely an engineering firm. The clarifications behind the lists were presented earlier, so those will be left out of this section.

5.7. Proposal and next steps

As this thesis was done and commissioned on the headquarters level of a diverse and large multinational, multi-industry organization it is difficult to define very concrete steps in the operational level. Nonetheless a proposal and some steps can be defined if on the higher level.

Overall it can be concluded that all of the three business models should be kept in mind as viable options going forward. The on-premises software needs to be an option for those moments when only that business model is due to obstacles possible. In those situations the on-premises software has significant benefits. Also the on-premises software needs to be kept as an option for those customer segments that are vary of new innovations. Software-based services should always be pursued not only on its own but also combined with the other two business models. The only pre-requisite is that services can be created from the outputs of the offered software. Software as a Service on the other hand is a very prominent business model that deserves a lot of attention. While this business model is only possible in the light of certain requirements the business model brings forth new opportunities that cannot be reached through the traditional on-premises software. A lot of these opportunities are data related and therefore especially interesting to an engineering firm whose expertise lies in understanding its products, processes and systems and therefore the data collected from those sources.

As the ABB does not yet possess the infrastructure to offer externally hosted delivery models in a wider extent the option of investing in the infrastructure or acquiring a partner for this purpose should be looked into and put into action. As soon as a suitable partner is found the infrastructure should be set up so that the actual plan to implement these business models can be put into action.

The first step to go forward is to set up a project group to further study the business models in a more detailed way taking into consideration all four components of the business model. The project should be started by testing the framework with already existing business models and modifying the framework accordingly. After this other possible business models should be come up with as a cohesive effort within the project. When looking into the possible new business models the opportunities brought up in this thesis should be taken into particular focus but at the same time the stated requirements need to be set in the background as prerequisites for each business model. These requirements cover both the requirements for the delivery models as well as the revenue models.

Simultaneously to using the framework to aid coming up with new business models additional studies must be done within the project to look into existing literature and especially already existing industrial software solutions in each business model type. External and internal benchmarking may bring forth more opportunities as well as restrictions to the business models. Also these studies should look into further analysis on the profitability of the different revenue models listed also in this thesis as well as cost analysis on the different business models that are come up with.

The duration of the project will be about six months during which a more in-depth knowledge on each business model will be acquired as well as a set of software business models, both internally and externally hosted will be documented.

From these business models use cases should be set up to test the framework as well as the business models in real situations. With the knowledge acquired from the studies done within the project and the use cases the framework presented in this thesis can be perfected to helping the business units find suitable business models from the documented set best fitted to their needs.

After the use cases more business models should be rolled out. Also the business units should be activated to start thinking of possible business models relevant to their business specifically. As the different industrial software business models have found a permanent place in the organizations product and service portfolios, the framework with slight modifications can be given to the sales organization in order to identify the best suited offering for each individual customer. This way supporting both reactive and proactive sales. It is important that the sales personnel learn the most important aspects of the business model so they can approach customers with the best fitting and most value bringing models.

Over all it would be important that the software business models, especially the new externally hosted software business models become a seamless part of the offering

portfolio of ABB. And thus helping ABB move on towards a more software and service oriented future.

6. CONCLUSIONS

This chapter rounds up the results of this study. First the academic contribution of this thesis is examined. The second chapter sums up the managerial implications of the study. The third chapter observes how well the objectives of this study were met and the fourth chapter discloses the limitations and some criticism in this study. Finally, possible future research possibilities that have arisen from this study are suggested in the last chapter.

6.1. Academic contribution

Different aspects of industrial software such as different architectures, for example by Koziolek et al. (2009), and quality, in literature by Choudhary (2007) for instance, have been studied in literature widely. Also the different software business models in regards to on-premises software and Software as a Service have been studied concerning the delivery models by Waters (2005), client side adoption by Xin & Levina (2008) and Benlian & Hess 2011) and revenue model options for example by Ojala (2013) and Konary et al. (2004), to name a few. They have also been somewhat thoroughly compared in previous literature such as in the articles by the previously mentioned Waters (2005) and Xin & Levina (2008). However, what the current literature is missing is connecting the industrial software setting to these business models and seeing what mode does the industrial setting bring to play in regards to the delivery and revenue models.

As the result of the thesis quite a few traits arose that have not been considered in the current literature. These aspects were brought up in connection to the type of software that would be offered in the industrial setting and by an industrial company. Most of these aspects had to do with the traits of the data that is used in the software, therefore, particularly focusing on the data-driven industrial software.

While the industrial setting is what can be regarded as the main academic contribution of the thesis, the thesis also brought to surface a new concept of software-based services that has not been introduced yet in any published studies. The concept in this thesis was derived from industry and introduced to answer the needs of the company, on behalf of which the thesis was conducted. Nonetheless, the concept is intriguing and could merit more research to be done in the future.

Another contribution of the thesis is the more comprehensive view on both the revenue models and the delivery models and the connections between them. Waters (2005), Foley (2004), Buxmann et al. (2008) and Benlian & Hess (2011) examine the business models, predominantly Software as a Service and on-premises software. The delivery models and the differences in the delivery models are quite well defined but the revenue model perspectives are very narrow. Limiting the on-premises software revenue model to licensing models and the Software as a Service revenue model as a utility or subscription models. Ojala (2013) compares the revenue models of the on-premises and Software as a Service models. However, Ojala (2013) does not go into detail about the delivery models. This thesis brings the two components of the business model tobether and establishes the distinctions between both and connects the two components together.

The framework presented in the thesis shows promise and it has passed the weak market test conducted through two workshops. The weak market test proved the suitability of the model for ABB on an abstract level but could be viewed as fitting to other similar industrial companies that are not seen primarily as a software company. However, the framework should be tested with actual existing business models and in various industries to be concluded as generally acceptable.

The fact that the empirical results are based on only the views of one company, makes the contribution somewhat one sided. However, the thesis can act as a stepping stone for future research, turning the attention towards engineering companies as the vendors of industrial software especially in regards to data driven offerings and the newer business models in software.

6.2. Meeting the objectives

This thesis concentrated on looking at three software based business models in an industrial setting and the distinctions between the three named business models. From the academic side the objective was to bring light on what differentiate the models from each other in an industrial setting. From ABB perspective the main objective was to look at the opportunities arising from the business models, especially Software as a Service and see if the business models should be further looked and invested into.

The main research question for this study was:

> *What aspects differentiate the alternative business models for industrial software service solutions?*

In addition three sub question were documented:

What is the nature of each business model: on-premises software, delivering software-based services and Software as a Service?

What are the things that an engineering firm needs to consider when choosing one of these business models?

What opportunities do the business models bring forth to an engineering firm?

The answer to the main research question is answered in chapter 4 where the results and therefore the main framework is presented. The framework assumes that the main differentiators for these business models are defined by aspects related to the target customer and value proposition ending up in different delivery and revenue models. These aspects are presented as questions in the framework.

The nature of each business model is explained in the literature review in chapters 2.3 to 2.6. These chapters explain the main nature of each business model on a rather abstract level without any connection to the industrial setting. As for the nature of the business models in an industrial setting, the explanations can be found in chapters 5.2, 5.3, 5.4. The chapters uncover the framework of this thesis in the perspective of each software business model. The answers to the two last sub questions that relate these business models to an engineering company are explained in chapters 5.2, 5.3 and 5.4 where the requirements and opportunities are examined per each business model. Appendix 5 concludes the requirements and opportunities of each business model in one table.

It can be stated that the study fulfilled the objectives set for it. While the topic still needs a lot of future research to be done, to fully answer the research questions in a more detailed way, in the set time and resource restrictions the study reached its objectives.

6.3. Managerial implications

ABB is looking ahead and the new trends in the software industry seem to bring new opportunities. Software has grown its importance in the corporate strategy and the company wants to get ahead of the emerging trends in the industrial software market. It was important to look into what the software business models could bring to an engineering company. Therefore, an initial comparison into the new and the traditional software business models had to be looked at in the industrial setting to see if the different software business models could bring anything to an engineering company.

The thesis shows that for an engineering company all three business models are indeed possible and beneficial for the engineering company in different scenarios. With the help of the framework the best suited business model for each scenario can be initially chosen. The biggest benefit though for the company is that the thesis acts as the pre-work for a larger project to look at different possible business models in the data driven

software marketplace. The framework and literature review will be a part of a further and more thorough analysis of different business models. The thesis presents the business models on a more abstract level without going into the value proposition options or target customers. This abstract presentation will act as the starting point for looking into more detailed business model option within the larger scope of the three business model categories: on-premises software, software-based services and Software as a Service.

The main focus of the project will be software-based services and Software as a Service with the data-driven services approach, while still touching on all three models overall. The project will start with an independent look into the models and the research on the topics. The view will be wider, by including all four components of the business model. Use cases and external case studies can be conducted to validate the revised framework. After this the framework and the check lists and the framework can be perfected according to the results of the literature and case studies done by the members of the research project. The framework should then be adapted for the use of sales personnel and the business units in order for the business units to themselves generate new business models in the scope of data-driven software services. This way the business models will be closest to each business unit's core competence and the business models will best serve the units and the targeted customers. The business models need to be documented and the information must always be communicated to sales so that the sales personnel always have an updated knowledge of the offering and can therefore match the offering to the customers in the best way possible.

The main implication to the company from this thesis is that it can be stated that all three business model types should be looked at. All of them have something to offer in different scenarios depending on the data, the environment and the targeted customer. Also the thesis brought up some opportunities of the externally hosted software offerings that should be further looked at and that can be utilized when coming up with more detailed business model plans.

6.4. Evaluation of the research

The thesis provided a way to find suitable software business models in the industrial setting. The limitations of this study result from the limitations in existing literature as well as the limitations from the empirical methods and the attempt to fit these two together systematically. These limitations though were recognized during the research and the effects were attempted to be minimized.

The literature review on the software business models was proven to be very lacking in the industrial scope. While this presented an opportune moment to bring something new to the academic scope of the topic, the fact that the company where the empirical study

was conducted also lacked experience in some of the business models, especially Software as a Service, also sets up a basis of quite significant limitations to the research. The results at some part are based on the perceptions and hypotheses of focus group participants and are very subjective to the focal organization as well as related experiences of the participants themselves. The participants to the empirical part of the study were chosen with a purpose to minimize the effects.

The participants had a very comprehensive knowledge of either all the industries that ABB operates in or a wide knowledge of a specific industry. The participants were also at least to an extent familiar with the software business models or even had some experience in the field. The main point though was that the participants were forward thinking and capable of looking at future possibilities instead of dwelling on things that already existed. This is also why the focus group discussions were chosen as a method for data collection. It served as the perfect platform for brainstorming on the idea and creating new perspectives. The participants, while sharing some similarities in their background, all had a specific knowledge and could therefore learn from each other and build on each other's ideas. As the discussions in the focus groups were free the ideas generated did not fall into too conservative limits as also very innovative ideas were explored. This brought new opportunities but at the same time the reoccurring subjects in different focus groups would deem to be valid.

The problem with the focus group discussions, especially since the discussions were unstructured to allow more space for imagination, is that the discussions could easily get off topic. Also, sometimes the discussions would focus on one specific topic on a very detailed level and therefore the participants would ignore the other topics at hand. This was attempted to be avoided by providing the discussion topics as a way to remind the participants would remember the entire scope of the discussions. In addition the moderator would remind the participants on the other topics by asking questions and trying to steer the conversation so that it would cover all topics in sufficient detail.

The focus group participants could not firsthand participate in the prioritizing and connecting the differentiating aspects that came up in the discussions. The connections were perceptions and interpretations done by the researcher. The effects of this was attempted to be minimized through the refining and validation rounds. Although as these already presented already visually constructed frameworks other options and interpretations may have been diminished through the visual interpretation.

The revising interviews were done individually in order to be able to concentrate on the specifics of the framework. If time had permitted it may have been beneficial to conduct more revising unstructured interviews. Then representatives of all business units and different levels of management in the organization could have been included. Now the revising interviews did not have as wide a scope as could have been possible. In order to

go around this problem people with a comprehensive knowledge of all the industries as well as software and research and development.

The validation round workshops gave a voice to the team that the thesis was conducted for and participants in the focus groups as well as the interviews. This way misunderstandings could be minimized and the framework validated. The first workshop for the Group Service team was deemed successful as there were no objections and a mutual agreement of the usability of the framework was made. The limitation, however, being that the team members are all not familiar with the topic and therefore might not have felt unqualified to comment due to lack of expertise. To go around this problem a comprehensive background on the business models was presented. The other workshop did not phase the same issue as the participants were familiar with the topics through the focus groups or interviews as well as form previous knowledge. However, the workshop was rather large and conducted as a conference call due to geographic dispersion. This may have unmotivated some people to participate in the discussion after the presentation. In order to minimize this problem questions were asked both directed to certain participants as well as for everyone simultaneously. The participants were also presented with the possibility to ask questions later on via email or phone if any questions or criticism arose after the workshop.

As explained the research was conducted as multi-method qualitative study by combining the previously mentioned methods. Each data collection method holds some limitations but also attempt to minimize the limitations of the previous method. This is the benefit of the multi-method study is that the subjectivity that stems from the interpretive study that the focus group discussions and interviews represent. While the multi-method aims to minimize the subjectivity, all the methods used are interpretations of the perceptions of the participants. And most importantly the participants in the focus groups and interviews represented all a single company. Therefore giving a very limited sample of the industrial field that was under examination. The fact that the thesis was conducted on behalf of this company gave a more comprehensive access to the experts within the organization it also limited the access to the external organizations. As the subject at parts, regarding the externally hosted software, is fairly new in the industry competitors are not willing to give their insight on the subject in fear of losing the first mover advantage and enabling the advancement of the competitors. Overall, even though the multi-method study minimized the subjectivity in the perspective of the individual participants by utilizing different qualitative methods. The methods are all subject to bias especially since the sample does not cover the entire industrial field and the opinions of the participants in each method may have been influenced by the shared corporate culture.

Another limitation that stemmed from the focus group and interview participants was that different people seemed to have different understandings on the business models.

Even though this was attempted to be minimized by providing a short basic information package on the business models, it cannot be said that misunderstandings and contradicting views on the basics of the business models were completely avoided and that they had no impact on the results. Perhaps the impact cannot be said to be the most significant but it must be taken into consideration when examining the results critically.

6.5. Future research

On-premises software business models is an old topic and quite widely researched in different angles from development to quality and architecture to name a few areas. Software as a Service is an emerging topic of research with more and more research being conducted. However in literature especially the latter is missing almost completely the industrial perspective. This is especially the case when it comes to non-software oriented engineering firms.

In the industry it can be seen that data-driven software and services is an emerging trend and as the topic is not available to be studied through literature reviews other methods should be considered. Case studies on companies already offering externally hosted software solutions and services could be conducted to find the industry norms and trends that are already identifiable in the industry. Software-based services is such a phenomenon that already exists in major engineering firms but has not been studied extensively in the academic field.

In order to find new opportunities that externally hosted software business models could bring to the industrial field and especially to the engineering firms as the vendors a mix of research methods may have to be conducted. To find more opportunities case studies on externally hosted software business models should be conducted focusing on the opportunities. These case studies should be done in software companies that offer Software as a Service in different industries. With input from these results an interview study could be conducted in engineering firms with R&D and software experts as well as industry experts. In fact a focus group study with a mix of these experts in each focus group would most probably bring the best perspective into the engineering company side of the study. If these focus groups or interviews can also be conducted in a variety of different engineering organizations the most comprehensive and valid results can be achieved.

The problem with advancing in this field on the industrial side is that Software as a Service and externally hosted software-based services are fairly new concepts in the industrial arena and, therefore, the organizations may be hesitant to sharing already conducted research on the topics in the fear of losing the first mover advantage in the field. Therefore, to get the best academic contribution the research should be done

independently from a single organization but rather try to achieve an industry wide perspective.

BIBLIOGRAPHY

Aisopos, F., Tserpes, K. & Varvarigou, T. 2013. Resource management in software as a service using the knapsack problem model. International Journal of Production Economics. Vol. 141(2), pp. 465–477.

Amit, R. & Zott, C. 2001. Value creation in e-business. Strategic Management Journal. Vol. 22(6-7), pp. 493–520.

Anderson, C. 2009. Free: How today's smartest businesses profit by giving something for nothing. 1st ed., London, Random House Business Books. 255 p.

Andris, R. P. 2002. Do Software Manufacturers Still Have the License to Sue ? Intellectual property & Technology Law Journal. Vol. 14(10), pp. 7–11.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. & Stoica, I. 2010. A View of Cloud Computing. Communications of the ACM. Vol. 53(4), pp. 50–58.

Benlian, A. & Hess, T. 2011. Opportunities and risks of software-as-a-service: Findings from a survey of IT executives. Decision Support Systems. Vol. 52(1), pp. 232–246.

Black, D. 2008. Building Better Software Better: Software-Enabled Services - Building Software That Powers Service. The Huffington post, pp. 1–37. [WWW]. Available at: http://big.assets.huffingtonpost.com/davidblack-paper.pdf. [Accessed: 5.10.2014]

Blanke, M., Izadi-Zamanabadi, R., Bogh, S. & Lunau, C. 1997. Fault-Tolerant Control Systems - A Holistic View. Control Engineering Practice. Vol. 5(5), pp. 693–702.

Boasson, M. 1993. Control Systems Software. IEEE Transactions on Automatic Control. Vol. 38(7), pp. 1094–1106.

Brooks-Harris, J. & Stock-Ward, S. 1999. Workshops: Desining and Facilitating Experimantal Learning. 1st ed., Thousand Oaks, California, SAGE Publication Inc. 188 p.

Bughin, J., Chui, M. & Manyika, J. 2010. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. McKinsey Quarterly. Vol. 56(1), pp. 75–86.

Butler, J. 1999. Risk Management Skills Needed in a Packaged Software Environment. Information Systems Management. Vol. 16(3), pp. 6–15.

Buxmann, P., Hess, T. & Lehmann, S. 2008. Software as a Service. Wirtschaftsinformatik. Vol. 50(6), pp. 500–503.

Casadesus-Masanell, R. & Ricart, J. 2010. From Strategy to Business Models and onto Tactics. Long Range Planning. Vol. 43(2-3), pp. 195–215.

Chesbrough, H. & Rosenbloom, R. 2002. The role of the business model in capturing value from innovation : evidence from Xerox Corporation's technology spin-off companies. Industrial and Corporate Change. Vol. 11(3), pp. 529–555.

Choudhary, V. 2007. Comparison of Software Quality Under Perpetual Licensing and Software as a Service. Journal of Management Information Systems. Vol. 24(2), pp. 141–165.

Corbin, J. & Strauss, A. 2008. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 3rd ed., Thousand Oaks, California, SAGE Publication Inc. 379 p.

Cusumano, M. 2008. The Changing Software Business : Moving from Products to Services. Computer. Vol. 41(1), pp. 20–27.

Cusumano, M. 2010. Cloud computing and SaaS as new computing platforms. Communications of the AMC. Vol. 53(4), pp. 27–29.

Dubey, A. & Wagle, D. 2007. Delivering software as a service. The McKinsey Quarterly. Vol. 6(2007), pp. 1–12.

Featherman, M. & Pavlou, P. 2003. Predicting e-services adoption: a perceived risk facets perspective. International Journal of Human-Computer Studies. Vol. 59(4), pp. 451–474.

Ferrante, D. 2006. Software Licensing Models : What's Out There? IT Pro. Vol. 8(6), pp. 24–29.

Ferrarini, L. & Carpanzano, E. 2002. A structured methodology for the design and implementation of control and supervision systems for robotic applications. IEEE Transactions on Control Systems Technology. Vol. 10(2), pp. 272–279.

Fisher, C. 2010. Researching and Writing a Dissertation: An essential Guide for Business Students. 3rd ed., Harlow, Essex, England, Pearson Education Limited. 437 p.

Foley, J. 2004. Power shift: Users gain flexibility and influence with new software pricing models. Information Week. Vol. 985(2004), pp. 47–48.

Greschler, D. & Mangan, T. 2002. Networking lessons in delivering "Software as a Service" - Part I. International Journal of Network Management. Vol. 12(5), pp. 317–321.

Hakkarainen, K. 2006. Strategic Management of Technology: From Creative Distruction to Superior Resilience. 162nd ed., Vaasa, University of Vaasa. 181 p.

Hennink, M., Hutter, I. & Bailey, A. 2011. Qualitative Research Methods. 1st ed., London, England, SAGE Publication Ltd. 300 p.

Jaffe, M., Leveson, N., Heimdahl, M. & Melhart, B. 1991. Software Requirements Analysis for Real-Time Process-Control Systems. IEEE Transactions on Software Engineering. Vol. 17(3), pp. 241–258.

Katzan, H. & Dowling, W. 2010. Software-As-A-Service Economics. The Review of Business Information Systems. Vol. 14(1), pp. 27–37.

Kettu, T., Kruse, E., Larsson, M. & Mustapic, G. 2008. Using Architecture Analysis to Evolve Complex Industrial Systems. Architecting Dependable Systems. Vol. 5(1), pp. 326–341.

Klenke, K. 2008. Qualitative Research in the Study of Leadership. 1st ed., Bingley, UK, Emerald Group Publishing Limited. 454 p.

Konary, A., Seymour, L. & Graham, S. 2004. The Future of Software Licensing : Software Licensing Under Siege. White paper: International Data Corporation, pp. 1–21.

Koziolek, H., Weiss, R. & Doppelhamer, J. 2009. Evolving Industrial Software Architectures into a Software Product Line: A Case Study. 5th International Conference on the Quality of Software Architectures, QoSA 2009 East Stroudsburg, PA, USA, June 24-26, pp. 177-193.

Krueger, R. & Casey, M. 2014. Focus groups: A Practical Guide for Applied Research. 5th ed., Thousand Oaks, California, SAGE Publication Inc. 280 p.

Lichter, H., Züllighcwen, H. & Schneider-Hufschrnidt, M. 1993. Prototyping in Industrial Software Projects - Bridging the Gap Between Software Projects Theory and Practice -. ICSE '93 Proceedings of the 15th international conference on Software Engineering. IEEE Computer Society Press Los Alamitos, CA, USA, pp. 221-229.

Van der Linden, F., Lundell, B. & Marttiin, P. 2009. Commodification of Industrial Software: A Case of Open Source. IEEE Software. Vol. 26(4), pp. 77–83.

Ma, D. 2007. The Business Model of "Software-As-A-Service." IEEE International Conference on Services Computing, IEEE, pp. 701-702.
Mason, J. 2006. Mixing methods in a qualitatively driven way. Qualitative Research. Vol. 6(1), pp. 9–25.
Morris, M., Schindehutte, M. & Allen, J. 2005. The entrepreneur's business model: toward a unified perspective. Journal of Business Research. Vol. 58(6), pp. 726–735.
Murphy, L. 2010. The Reality of Freemium in SaaS. Sixteen Ventures, p. 25.
Ojala, A. 2013. Software-as-a- Service Revenue models. IT Pro. Vol. 15(3), pp. 54–60.
Ojala, A. & Tyrväinen, P. 2012. Revenue Models in Cloud Computing. 2nd Annual International Conference on Computer Games, Multimedia and Allied Technology (CGAT 2012), Global Science & Technology Forum (GSTF), Singapore, pp. 114-119.
Osterwalder, A. & Pigneur, Y. 2010. Business Model Generation: A Handbook For Visionaries, Game Changers, And Challengers. Hoboken, New Jersey, John Wiley & Sons Inc. 288 p.
Ostrand, T. & Weyuker, E. 2002. The distribution of faults in a large industrial software system. AMC SIGSOFT Software engineering notes. Vol. 27(4), pp. 55–64.
Rust, R. & Kannan, P. 2002. E-Service: New directions in theory and service. Armonk, New York, USA, M.E. Sharpe. 239 p.
Rust, R. & Kannan, P. 2003. E-Service: A new paradigm for business in the electronic environment. Communications of the AMC. Vol. 46(6), pp. 36–42.
Rust, R. & Lemon, K. 2001. E-Service and the Consumer. International Journal of Electronic Commerce. Vol. 5(3), pp. 85–101.
Sainio, L. & Marjakoski, E. 2009. The logic of revenue logic? Strategic and operational levels of pricing in the context of software business. Technovation. Vol. 29(5), pp. 368–378.
Saunders, M., Lewis, P. & Thornhill, A. 2009. Research Methods for Business Students. 5th ed., Essex, England, Pearson Education Limited. 627 p.
Schnjakin, M., Alnemr, R. & Meinel, C. 2010. A Security and High-Availability Layer for Cloud Storage. 1st International Workshop on Cloud Information System Engineering, Hong Kong, China, December, pp. 449-462.
Shafer, S., Smith, H. & Linder, J. 2005. The power of business models. Business Horizons. Vol. 48(3), pp. 199–207.
Shapiro, C. & Varian, H. 1998. Versioning: the smart way to sell information. Harward Business Review. Vol. 76(6), pp. 107–114.
Sharpe, S. & Nguyen, H. 1995. Capital market imperfections and the incentive to lease. Journal of Financial Economics. Vol. 39(2-3), pp. 271–294.
Silverman, D. 2013. Doing Qualitative Research: A Practical Handbook. 4th ed., London, Englan, SAGE Publication Ltd. 488 p.
Smedinghoff, T. 2008. Information Security Law: The Emerging Standard for Corporate Compliance. Ely, UK, IT Governance Pub. 175 p.
Stewart, D. & Shamdasani, P. 2014. Focus Groups: Theory and Practice. 3rd ed., Thousand Oaks, California, SAGE Publication Inc. 224 p.
Sun, W., Zhang, X., Guo, C., Sun, P. & Su, H. 2008. Software as a Service: Configuration and Customization Perspectives. Congress on Services Part II, IEEE Computer Society, pp.18-25.

Tashakkori, A. & Teddlie, C. 2010. SAGE Handbook of Mixed methods in Social & Behavioral Research. 2nd ed., Thousand Oaks, California, SAGE Publication Inc. 822 p.

Teece, D. 2010. Business Models, Business Strategy and Innovation. Long Range Planning. Vol. 43(2-3), pp. 172–194.

Tsvetkova, A. & Gustafsson, M. 2012. Business models for industrial ecosystems: a modular approach. Journal of Cleaner Production. Vol. 29-30, pp. 246–254.

Waters, B. 2005. Software as a service: A look at the customer benefits. Journal of Digital Asset Management. Vol. 1(1), pp. 32–39.

Weil, N. 2007. Get Smart About Saas ; Vendors say software as a service will cut costs and increase efficiency. They say it's enterprise ready. Does that sound too good to be true? It is. CIO. Vol. 20(16), pp. 1–6.

Weinhardt, C., Anandasivam, A., Blau, B. & Stößer, J. 2009. Business Models in the Service World. IT Professional. Vol. 11(2), pp. 28–33.

Wilson, F. 2006. The Freemium Business Model. VC & Technology, p. 1. [WWW]. Available at: avc.com/2006/the_freemium_bu/. [Accessed: 29.9.2014].

Xin, M. & Levina, N. 2008. Software-as-a-Service Model: Elaborating Client-side Adoption Factors. Proceedings of the 29th International Conference on Information Systems, Paris, France, December 14-17, pp. 1-12.

APPENDICES (5 pieces)

APPENDIX 1: Background on the industrial software service solutions

Industrial software service solutions

All Software solutions described in this scope are related to and should have an impact on the service business.

Licensing Software

- The customer buys a permanent license to a software application.
- The software is installed, run and maintained on the customer's hardware.
- The licensing requires a single upfront payment for the license to use the software and possible future payments for upgrades
- Requires the customer to make additional investments to hardware, installation and maintenance (other possible hidden costs for customer)

Software as a Service

- Based on multi-tenant architecture which means that there is only one common code that runs in the vendor's server
- Customer accesses software online
- Vendor responsible for maintenance of software and storing customer's data
- Subscription or pay-per-use model – customer only pays for use of software
- customer still has administrative control, customization on metadata level
- supplier more control on the future development of the application
- All upgrades automatically and instantly available to the customer

Software-based services

- The internal software produces data directly used to deliver specific services
- Customer has no access to the software
- e.g. energy analysis

APPENDIX 2: Discussion topics for focus group interviews

Focus group interview

The aim of this interview is to come up possible success factors of different business models related to industrial software service solutions and what benefits the business models may bring to ABB. Also this interview will be the basis of distinguishing what criteria a business model should fulfill for it to fit to the ABB portfolio and what benefits and aspects are required and what restrictions there may be.

Discussion topics:

- What is your core business model?
- How do you handle your software lifecycle
- Success factors
- Service aspect
- Revenue model of the BM

Industrial software service solutions

- Success factors and benefits overall

Licensed Software

- What could Licensing Software bring to ABB
- Possible business models
- Revenue models
- Success factors
- Benefits
- Pitfalls
- Experiences
    - What worked
    - What did not work

**

Software as a Service

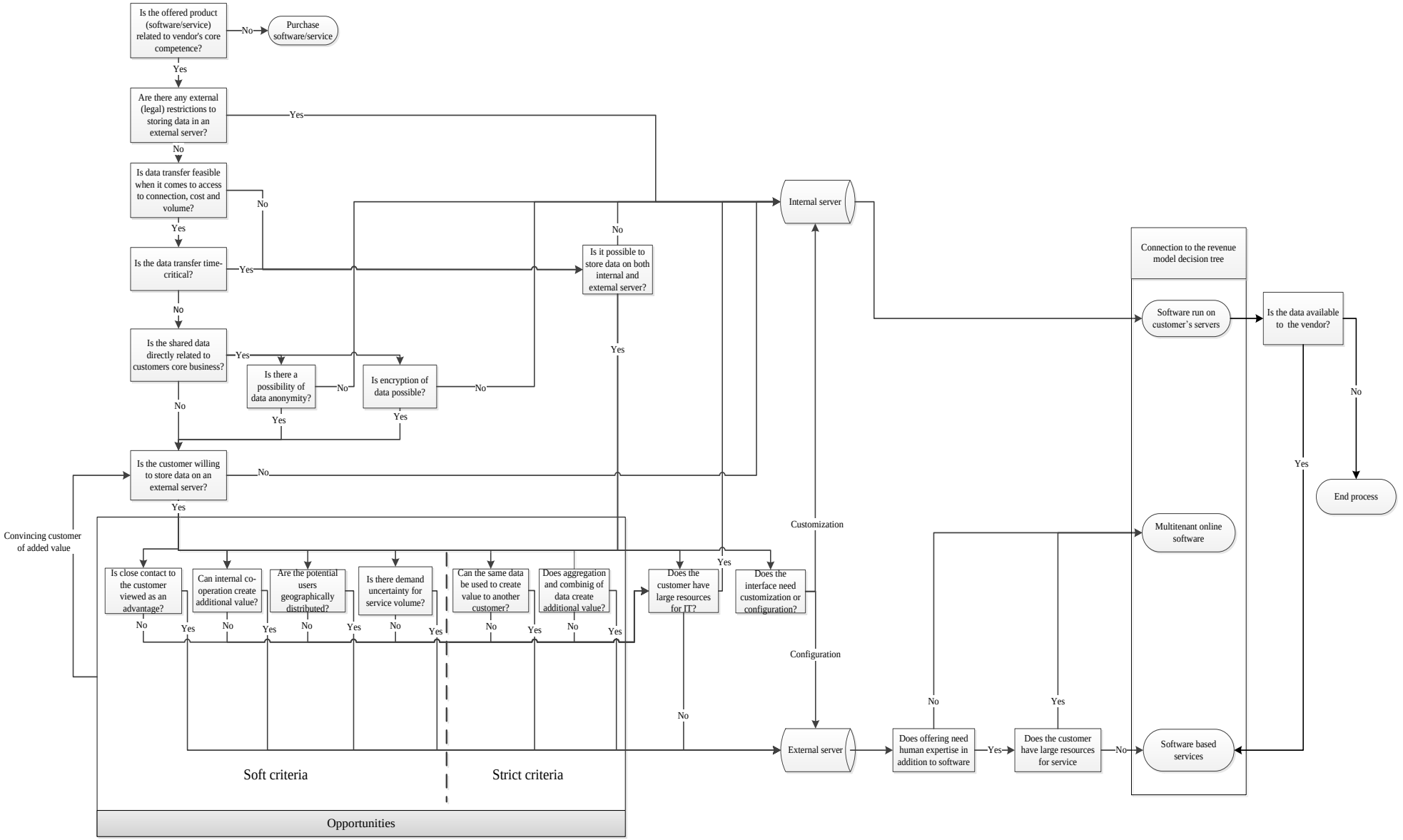
- What could SaaS bring to ABB
- Possible business models
- Revenue models
- Success factors
- Benefits
- Pitfalls
- Experiences
    - What worked
    - What did not work

Software-based services

- What could Service based on internal software bring to ABB
- Possible business models
- Revenue models
- Success factors
- Benefits
- Pitfalls
- Experiences
    - What worked
    - What did not work

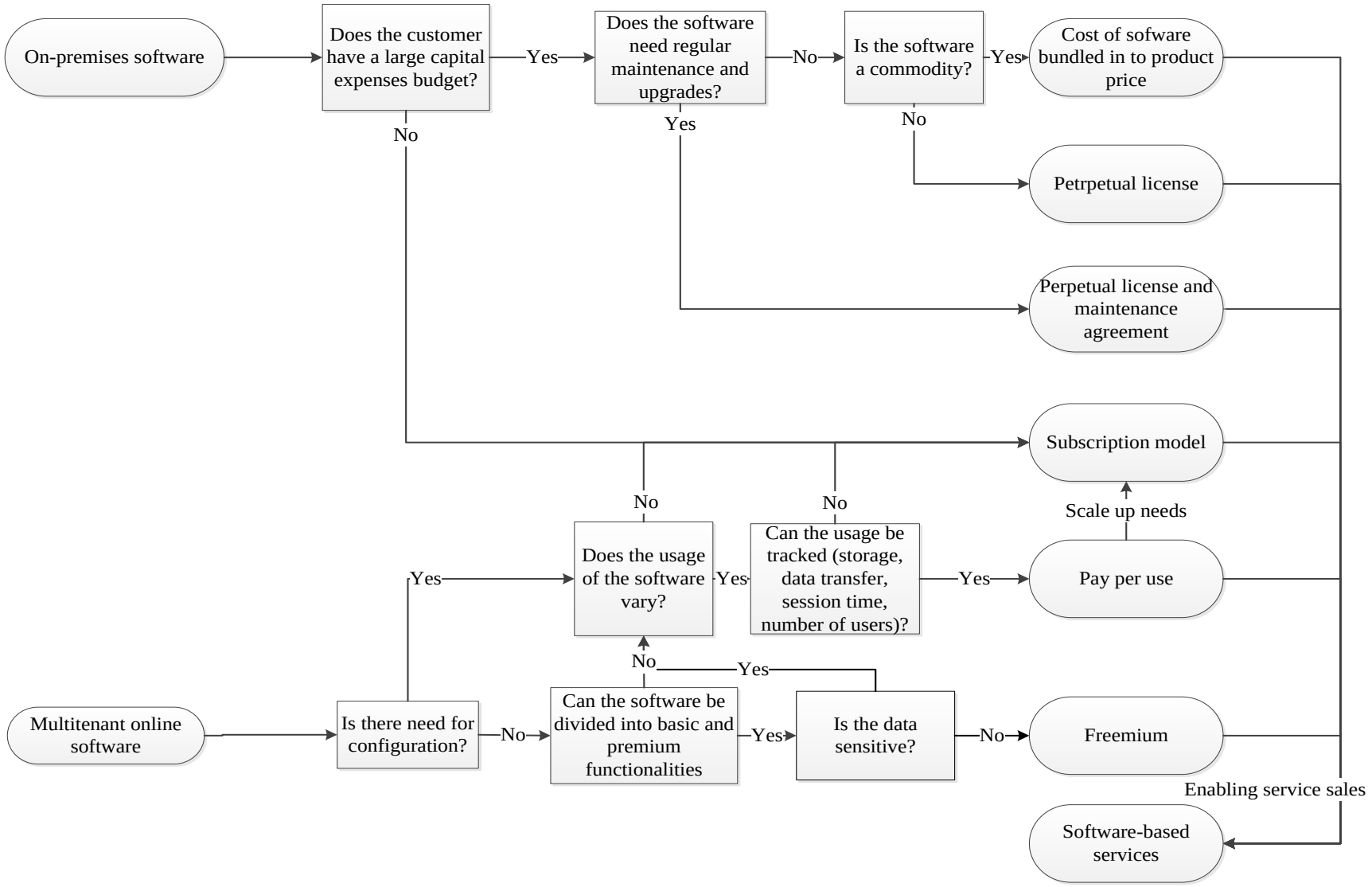
**

APPENDIX 3: Delivery model decision tree



**

APPENDIX 4: Revenue model decision tree



**

APPENDIX 5: Requirements and opportunities of the industrial software business models

	On-premises software	Software-based services	Software as a Service
Requirements	• The customer must have sufficient IT resources and infrastructure to host the software application	• The outputs of the software must be available • The outputs can be converted into services	• There cannot be any legal restrictions to storing data on external servers • The data cannot be time critical o If the data is time critical there should be a possibility to store the data both internally and externally • The data should not be related to the customers core business o Unless the data can be encrypted or anonymity ensured • The customer must give consent to storing data on the external servers
Opportunities	• There are usually no external restrictions to storing the data on the customer's servers. • The software can be run offline • Any kind of data can be stored on the customer's servers regardless of the time-critical or sensitive nature of the data • Customers trust the internal servers easier • If the customer has large IT resources they may be more inclined to adopting software running on their own servers • If the software needs customization on-premises software is the less costly option	•Opportunities of software-based services when using the external server: • Internal co-operation has a more easily accessible platform • The data can be used and converted to serve the needs of other target customers • Data aggregation can create additional value and new opportunities	• Close contact to the customer is easy to achieve • Internal co-operation has a more easily accessible platform • Adopting the software is more beneficial for the customer if users are geographically distributed • Demand uncertainty can be contained better • The data can be used and converted to serve the needs of other target customers • Data aggregation can create additional value and new opportunities